Vogliamo
Incominciare
Cosí ?

IMPARIAMO A PROGRAMMARE IN BASIC CON IL VIC™/CBM™

Rita Bonelli

GRUPPO EDITORIALE JACKSON

Vogliamo
Incominciare
Cosí ?

IMPARIAMO A PROGRAMMARE IN BASIC CON IL VIC™/CBM™

di
Rita Bonelli

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

A Luigi Bonezzi
sempre vivo nel nostro ricordo

L'autrice ringrazia i signori Maurizio Guardassoni, Salvatore Riefoli e Giovanni Valerio per la costruttiva collaborazione.

Il Gruppo Editoriale Jackson ringrazia per il prezioso lavoro svolto nella stesura del volume le signore Francesca di Fiore, Rosi Bozzolo e l'ing. Roberto Pancaldi.

Seconda edizione: 1982

Stampato in Italia da:
S.p.A. Alberto Matarelli - Milano - Stabilimento Grafico

P R E F A Z I O N E

Persino nell'impostazione grafica il titolo di questa nuova fatica della dr. Rita Bonelli evidenzia l'acronimo del VIC, il personal di basso costo che la Commodore ha lanciato sul mercato di quella che ormai puo' essere definita informatica di massa. Ma ancor piu' felice a noi sembra la semantica: "Vogliamo Incominciare Cosi'?" mette infatti subito in evidenza che il testo si rivolge ai principianti e lo fa in forma dubitativa e propositiva insieme. E c'e' un preciso motivo in tutto cio'. Prima di chiarirlo e' opportuno ricordare che questa e' praticamente la quinta opera che l'Autrice riesce a mettere in cantiere nel giro di una diecina di mesi. Infatti ai precedenti testi, pubblicati per questi medesimi tipi jacksoniani, il primo relativo allo ZX-80 e l'altro che insegna a programmare in BASIC sul personal computer PET, vanno aggiunti due manuali operativi, l'uno sul microcomputer DAI e il secondo sempre per lo ZX-80, attrezzato con la nuova ROM. Tanta prolificita' si spiega anzitutto con capacita' invidiabili di resistenza alla fatica e di organizzazione razionale del proprio tempo ma non nasce certo per caso: gia' nelle precedenti edizioni abbiamo avuto occasione di far risaltare come essa sia il frutto di grande esperienza professionale e, insieme, didattica. Una combinazione abbastanza rara in una medesima persona, corroborata da una notevole sensibilita' verso il fenomeno nuovo della microinformatica e delle sue connotazioni diremo psico-pedagociche. Ci spieghiamo: e' abbastanza evidente che la "fame" di testi come questo nasce anche da difficolta' che la nuova utenza incontra nell'accostarsi alle prodigiose macchinette, che finiscono per essere (o apparire: ma e' lo stesso...), oggetto di un evidente complesso freudiano di odio-amore.

A questi "amanti" (spesso) delusi la Bonelli nei suoi precedenti manuali ha fatto in sostanza una proposta, per cosi' dire, "rigorosa": venivano cioe' premessi dei capitoli di background, quali richiami sulla diagrammazione a blocchi e sui concetti informatici di base. Questo evitava il rischio dell'aridita' di un approccio troppo tecnicistico ed esclusivamente legato alla particolare macchina. Ebbene stavolta questo approccio e', almeno formalmente, capovolto. L'impostazione e' molto piu' pratica e si parte, fin dall'inizio, con esempi d'utilizzo del VIC. Non ci sembra che si possa trovare, a priori, nulla di scandaloso in questa, apparente, contraddizione. La coerenza - e' stato detto - e' l'unica virtu' degli spiriti mediocri. Stavolta l'Autrice si e' soprattutto preoccupata di quei problemi di tipo psicologico ai quali sopra accennavamo e, dietro questa impostazione apparentemente a-sistematica c'e' in realta' lo

sforzo di chi vuole conseguire una graduale, amichevole familiarizzazione dell'utente-neofita con la macchina, vincendo le inevitabili, anche se inconsce, diffidenze e gli oscuri timori che forse albergano in tutti noi verso le nuove diavolerie tecnologiche.

Se si guarda bene, ci sembra allora che non ci sia contraddizione.

D'altronde quello tra spontaneismo sperimentale e metodicita' e' un tormento che assilla da sempre la didattica: l'importante e' operare una sintesi tra opposte esigenze, entrambe corrette, facendola nel modo giusto al momento giusto e una volta che si sono definiti gli obiettivi.

Ecco infine perche' siamo sicuri che anche questo libro sara' un successo: l'onesta' della proposta (espressa dal punto di domanda) nasce dal vivo di un'esperienza didattica ed avra' echi favorevoli non solo presso utenti-discenti ma -contiamo- anche tra tutti coloro che fanno questo ingrato mestiere di operatore scolastico.

Gianni Giaccaglini

S O M M A R I O

CAPITOLO 1

I N T R O D U Z I O N E

1.1. PRELIMINARI

Lo scopo di questo manuale e' quello di iniziare i principianti alla programmazione in linguaggio BASIC del calcolatore elettronico VIC 20.

Vediamo subito il significato del nome VIC; sono le iniziali di Video Interface Computer, si tratta cioe' di un calcolatore collegabile a un qualunque televisore.

Tra gli acquirenti del VIC ci saranno sicuramente anche degli esperti programmatori, per queste persone sara' sufficiente dare una rapida scorsa ai punti del manuale che mettono in evidenza le caratteristiche di questo nuovo calcolatore, e poi sapranno da soli quello che devono fare.

Si tenga presente che con questo manuale non si pretende di dire tutto sul VIC, si desidera solo cominciare a parlare di questo nuovo e molto interessante calcolatore della COMMODORE, appena arrivato in Italia. E' possibile quindi che a questo primo manuale faccia seguito un secondo per completare gli argomenti non esauriti e per parlare ex novo di quelli qui solo accennati.
In questo manuale si trovano solo dei cenni alle periferiche come disco e stampante, al trattamento dei files, al multicolore, alle funzioni assegnabili, alle schede ROM aggiuntive, alla espansione della memoria RAM, alla programmazione in linguaggio macchina, al sistema operativo, alle periferiche speciali.

I principianti dovrebbero cominciare la lettura del manuale dall'inizio e seguire tutti i consigli che vengono dati. Per non spaventare i lettori con una rigorosa teoria, si e' cercato di esporre gli argomenti eseguendo degli esercizi, in tale modo quando si espone la teoria la pratica dovrebbe gia' averla fatta acquisire almeno in parte.

Leggendo i primi 5 capitoli il lettore dovrebbe, speriamo divertendosi, prendere confidenza con le possibilita' del VIC, anche quelle grafiche e sonore, e nel frattempo imparare praticamente che cosa e' un programma. In questi capitoli sono contenuti richiami alle Appendici che sono fondamentali per una buona conoscenza del calcolatore.

Nel capitolo 6 si spiega a grandi blocchi come funziona il VIC e nel capitolo 7 si cerca di entrare piu' sistematicamente nell'argomento "programmazione".

Si raccomanda di non scoraggiarsi alle prime difficolta', imparare a programmare un calcolatore elettronico non e' difficile, ma ci vuole un po' di pazienza e non si deve avere fretta.

1.2. COME INSTALLARE IL VIC

Quando tornate a casa con il vostro VIC cercate un posto tranquillo dove poter lavorare, possibilmente con un tavolo vicino a una presa di corrente.

Aprite la confezione estraete il calcolatore, che apparentemente e' solo una tastiera e collocatelo sul tavolo; poi estraete l'alimentatore, scatola con 2 cavi alle estremita', e ponetelo dietro il calcolatore dalla parte destra.

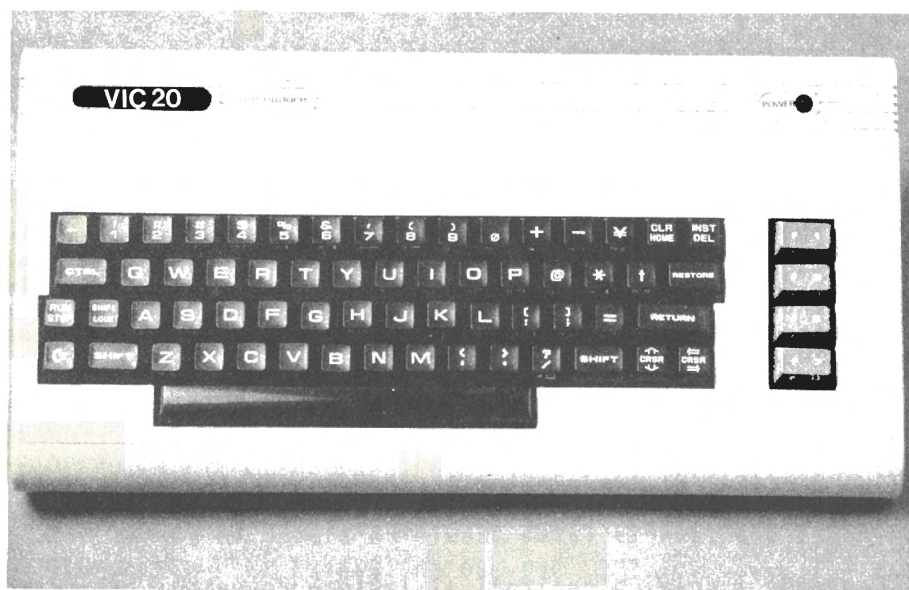



Figura 1-1. Il VIC 20.

Avrete gia' predisposto un televisore da collegare al calcolatore, ponetelo sul tavolo alla sinistra del VIC.

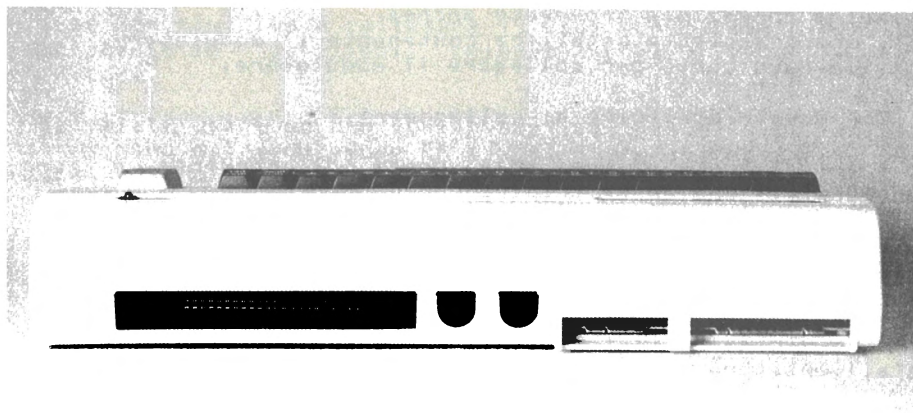



Figura 1-2. Connessioni posteriori.

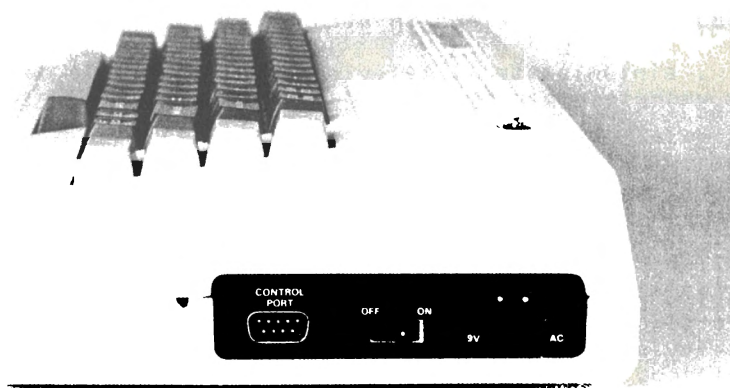



Figura 1-3. Connessioni laterali.

Nella confezione troverete anche:
. una scatoletta metallica contenente il modulatore;
. un cavo corto per collegare il modulatore.

Ora dovete procedere ai collegamenti:

. se disponete di una sola presa di corrente a muro, procuratevi una presa tripla;

. assicuratevi che l'interruttore ON/OFF di accensione del VIC posto sul lato destro sia in posizione OFF;

. collegate l'alimentatore alla presa 9V AC posta sul lato destro vicino all'interruttore di accensione e poi inserite l'alimentatore nella presa di corrente;

. collegate il cavo corto nella prima presa (VIDEO PORT) posta sul retro del VIC nella parte centrale a sinistra vicino alla fessura grande (state attenti di non usare la presa sulla destra che e' una porta seriale 6 PIN, tra l'altro lo spinotto non potrebbe entrarvi);

. collegate il modulatore RF al televisore (usando un cacciavite se manca lo spinotto) all'entrata dell'antenna VHF:

. accendete il televisore;

. accendete il VIC e vedrete accendersi la spia rossa posta sulla tastiera in alto a destra;

. sintonizzate il televisore sul canale 36 della gamma UHF e vedrete apparire sul video quanto segue:

```
**** CBM BASIC V2 ****

3583 BYTES FREE

READY.
```

sullo schermo dovreste vedere un bordo che circonda un rettangolo di colori diversi; i colori dipendono dal televisore, ma dovrebbe essere il bordo azzurro e lo schermo bianco. Inoltre il numero 3583 dipende dalla memoria RAM del calcolatore e quindi potrebbe essere diverso;

. se non arrivate a questo risultato spegnete il VIC e riprovate;

. ponete il volume del televisore al minimo per non essere disturbati da rumore;

. se non riuscite a collegare il VIC, pur avendo seguito esattamente le istruzioni, chiedete consiglio al vostro fornitore.

Se volete usare un Monitor invece del televisore potete fare a meno del modulatore RF e collegare direttamente il VIC al vostro Monitor.

1.3. PANORAMICA SUL VIC

Ora siete seduti davanti al vostro VIC, avete acceso e sintonizzato il televisore ed avete acceso il calcolatore. Il video si presenta con un bordo in azzurro che circonda un quadro in bianco. Nel riquadro compaiono delle scritte in blu che riportiamo:

```
 ----------------------
!**** CBM BASIC V2 ****!
!3583 BYTES FREE       !
!READY.                !
![]  <------------------------- quadratino
!                      ! blu lampeggiante
!                      ! che verra' chia=
!                      ! mato CURSORE nel
!                      ! seguito.
 ----------------------
```

Sul significato delle prime 2 linee di scrittura torneremo in seguito. Il numero che compare all'inizio della seconda linea, prima della parola BYTES, puo' anche essere diverso da 3583 e dipende dalla grandezza della memoria del calcolatore, nella terza linea la parola READY sta a significare che il VIC e' pronto a dialogare con voi. Il cursore lampeggiante nella quarta riga indica il punto dello schermo dove apparira' il primo carattere che voi premerete sulla tastiera. Nel seguito useremo "[]" per indicare la posizione del CURSORE sul video.

Provate a digitare:

    STO IMPARANDO AD USARE IL VIC

e quando avete digitato l'ultimo carattere (C) premete il tasto rettangolare recante la scritta RETURN. Le scritte che gia' apparivano sul video si sono ora arrichite cosi':

```
**** CBM BASIC V2 ****
3583 BYTES FREE
READY.
STO IMPARANDO AD USARE
 IL VIC

?SYNTAX
 ERRROR
READY.
[]
```

Come potete vedere, appare ancora READY e il cursore lampeggiante sulla riga sottostante. Perche' appare: ?SYNTAX ERROR ?

Quando voi vedete la scritta READY il calcolatore e' pronto a dialogare con voi, ma nel linguaggio a lui gradito, che e' il linguaggio BASIC. Questo linguaggio pretende che le vostre frasi siano scritte rispettando alcune regole formali che imparerete a poco a poco.

Ora spegnete il calcolatore e riaccendetelo subito dopo. Rivedrete sullo schermo la scritta iniziale. Ora digitate ordinatamente quanto segue, premendo il tasto RETURN alla fine di ogni linea indicata:

```
10PRINT"STO IMPARANDO AD USARE IL VIC ";
20GOTO10
```

Ora lo schermo si presenta cosi':

```
**** CBM BASIC V2 ****
3583 BYTES FREE
READY.
10PRINT"STO IMPARANDO
AD USARE IL VIC ";
20GOTO10
[]
```

A questo punto digitate RUN e poi premete RETURN voi vedrete che lo schermo viene riempito con continuita' dalla scritta che voi avete digitato tra gli apici: STO IMPARANDO AD USARE IL VIC, e che quando lo schermo e' pieno le scritte risalgono con continuita' verso l'alto con notevole velocita'. Questa velocita' e' tale che riesce difficile leggere le parole. Provate a tenere premuto il tasto CTRL (primo a sinistra della seconda riga di tasti dall'alto), e vedrete che la velocita' di movimento delle scritte sullo schermo rallenta. Se provate ora a premere il tasto RUN/STOP (primo a sinistra della terza riga di tasti dall'alto), vedrete che le scritte si fermano, si liberano alcune righe verso il basso e compare la scritta:

```
BREAK IN 10
READY.
[]
```

Provate ora a digitare CONT e poi premete il tasto RETURN. Vedrete che le scritte ricominciano a muoversi sullo schermo.
Senza rendervene conto avete scritto il vostro primo programma in BASIC e lo avete "fatto girare".

Se avete incontrato delle difficolta' nel seguire i suggerimenti del manuale e cioe' non avete ottenuto i risultati voluti, probabilmente non avete eseguito correttamente quanto esposto. Dovete allora spegnere il calcolatore, attendere qualche secondo, riaccenderlo e ricominciare da capo. Ricordatevi che non e' molto difficile imparare ad usare un calcolatore elettronico, ma ci vuole un po' di pazienza e bisogna cercare di essere precisi.

Rivediamo quello che e' successo:

- all'inizio abbiamo dialogato con il calcolatore non in BASIC ed il calcolatore ci ha avvisato che avevamo fatto un errore di sintassi;
- dopo abbiamo scritto un programma BASIC formato da due frasi che iniziavano con i numeri 10 e 20 e terminavano con l'uso del tasto RETURN;
- abbiamo notato che se mentre scriviamo qualcosa la linea dello schermo si riempie il calcolatore va a capo in mo= do automatico;
- abbiamo fatto girare il programma scrivendo RUN e poi RETURN;
- abbiamo interrotto il programma che "girava" premendo il tasto RUN/STOP;
- abbiamo fatto continuare il programma interrotto digi= tando CONT e premendo RETURN;
- abbiamo notato che se si continua a scrivere quando lo schermo e' pieno le linee piu' in alto spariscono e si liberano delle linee in basso, si ha cioe' quello che viene denominato "SCROLLING";
- abbiamo visto che tenendo premuto il tasto CTRL si ha il rallentamento dello scrolling;
- abbiamo visto che le scritte compaiono solo nel riquadro dello schermo e mai sul bordo.

Possiamo dire, dopo questa piccola prova, che il nostro VIC ci consente di dialogare con lui in modo abbastanza semplice, e che la tastiera e lo schermo del video sono collegati tra loro. In seguito capirete meglio tutto quello che avete visto accadere in questa prima prova.

CAPITOLO 2

L A T A S T I E R A E IL VIDEO

2.1. LA TASTIERA DEL VIC

A questo punto dobbiamo prendere confidenza con la tastiera del VIC. Osservando i tasti possiamo notare che alcuni recano solo qualcosa impresso sopra, mentre molti altri recano anche delle scritte o due caratteri grafici impressi sulla parte frontale. Possiamo dedurre che mentre alcuni tasti svolgono una sola funzione, altri possono svolgere piu' funzioni se usati in particolari condizioni.

Accendete il vostro calcolatore ed osservate la solita scritta che compare sul video. Vedete tutti caratteri maiuscoli. Ora provate a premere contemporaneamente i due tasti in basso a sinistra contrassegnati rispettivamente dal marchio della COMMODORE e da SHIFT, vedrete che la scritta sul video diventa tutta in caratteri minuscoli, mentre gli asterischi ed il numero non variano. Provate diverse volte a premere contemporaneamente questi due tasti e vedrete cambiare il contenuto del video. Questo comportamento dipende dal fatto che il calcolatore dispone di due gruppi di caratteri tra loro indipendenti; al momento dell'accensione del calcolatore e' attivo il gruppo che scrive le lettere maiuscole. Se si premono contemporaneamente i due tasti COMMODORE e SHIFT diventa attivo il secondo gruppo di caratteri e questo fatto si riflette immediatamente sul video. I gruppi di caratteri vengono anche chiamati "SET". Il primo gruppo viene chiamato SET GRAFICO, il secondo SET MAIUSCOLO/MINUSCOLO. Quest'ultimo e' molto simile al gruppo di caratteri di una normale macchina da scrivere.

Cominciamo a descrivere i tasti che hanno lo stesso significato nei due set di caratteri.

Il tasto largo in basso e' quello dello spazio, come sulle normali macchine da scrivere. Esso ha inoltre la caratteristica che se si tiene premuto ripete lo spazio, cioe' ha la funzione di "RIPETIZIONE" automatica.

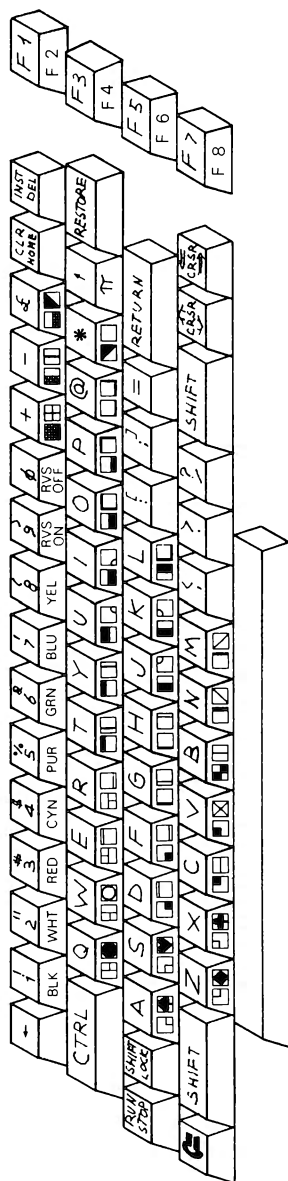



Figura 2-1. La tastiera del VIC.

Esistono due tasti marcati SHIFT (uno a destra ed uno a sinistra). Le funzioni di questi tasti sono quelle di modificare il normale comportamento di un tasto se premuti contemporaneamente ad esso (come per le maiuscole nelle normali macchine da scrivere). Esiste anche un tasto SHIFT LOCK che serve per bloccare i tasti SHIFT.

Il tasto RETURN serve per avvisare il calcolatore che si e' terminato di scrivere, cioe' fa accettare al calcolatore quello che si e' appena scritto.

Il tasto CLR HOME serve per mandare il cursore dello schermo nell'angolo in alto a sinistra. Se premete questo tasto contemporaneamente allo SHIFT ottenete di ripulire lo schermo e di mandare il cursore in alto a sinistra.

Il tasto INST DEL serve per cancellare l'ultimo carattere scritto, cioe' fa tornare indietro il cursore di una posizione. Se usato insieme allo SHIFT serve per creare uno spazio per inserire un nuovo carattere. Anche questo tasto ha la funzione di ripetizione automatica se si mantiene premuto.

Il tasto CTRL serve per rallentare il movimento dello schermo durante lo scrolling. Esso serve inoltre, in modo simile allo SHIFT, per attivare particolari funzioni.

I TASTI COLORE sono quelli che recano sulla parte frontale le scritte:

BLK WHT RED CYN PUR GRN BLU YEL

essi si usano tenendo contemporaneamente premuto il tasto CTRL e servono per modificare il colore dei caratteri che si scrivono sullo schermo. Se premete CTRL e contemporaneamente il tasto RED, il colore del cursore diventa rosso, se ora scrivete qualcosa lo vedrete apparire in rosso invece che in blu.

Il tasto che reca sulla parte frontale RVS ON se premuto insieme a CTRL modifica il modo di scrittura sullo schermo, cioe' il carattere appare in campo inverso. Invece di vedere, per esempio, un carattere blu su sfondo bianco, si vede un carattere bianco entro un rettangolino blu.

Il tasto che reca sulla parte frontale RVS OFF se premuto insieme a CTRL modifica il modo di scrittura sullo schermo, cioe' fa ritornare dal campo inverso al modo normale.

Il tasto RESTORE se premuto insieme al tasto RUN STOP serve per riportare il calcolatore allo stesso stato di quando viene acceso, ma con la differenza che il programma

eventualmente presente in memoria non viene cancellato.

Il tasto RUN STOP, oltre alla funzione vista sopra, serve per fermare un programma che sta girando. Se si usa questo tasto insieme a SHIFT esso serve per caricare in memoria dal nastro del registratore il primo programma che si incontra.

Il tasto CRSR con le due frecce alto/basso serve per far muovere il cursore sullo schermo verso il basso. Se usato insieme a SHIFT fa muovere il cursore verso l'alto. Questo tasto ha la funzione di ripetizione automatica.

Il tasto CRSR con le due frecce sinistra/destra serve per far muovere il cursore sullo schermo verso destra. Se usato insieme a SHIFT fa muovere il cursore verso sinistra. Questo tasto ha la funzione di ripetizione automatica.

I 4 tasti a destra con le scritte: F1 e F2, F3 e F4, F5 e F6, F7 e F8 servono per eseguire funzioni speciali. Usati da soli rendono attiva la funzione scritta sopra, mentre usati insieme a SHIFT rendono attiva la funzione scritta sulla parte frontale.

Consideriamo ora il SET GRAFICO. Usando i tasti che recano un solo carattere sulla parte superiore ottenete quel carattere, se esso e' una lettera la vedete maiuscola. Se i tasti recano due caratteri sulla parte superiore ottenete quello scritto piu' in basso. Per ottenere quello scritto piu' in alto dovete usare contemporaneamente il tasto SHIFT. Per ottenere i caratteri grafici, scritti sulla parte frontale, dovete procedere cosi':

- premendo il tasto insieme allo SHIFT si ottiene il carattere grafico di destra;
- premendo il tasto insieme al tasto COMMODORE si ottiene il carattere grafico di sinistra.

Consideriamo ora il SET MAIUSCOLO/MINUSCOLO. Usando i tasti che recano un solo carattere sulla parte superiore ottenete quel carattere, e se esso e' una lettera la vedete minuscola. Se i tasti recano due caratteri sulla parte superiore ottenete quello scritto piu' in basso. Per ottenere quello scritto piu' in alto dovete premere contemporaneamente il tasto SHIFT. I tasti con le lettere dell'alfabeto se usati insieme a SHIFT danno le lettere maiuscole, se usati insieme al tasto COMMODORE danno il carattere grafico di sinistra, salvo alcuni per i quali il carattere grafico non corrisponde. Per gli altri tasti (quelli che non hanno una lettera sulla parte superiore) i caratteri grafici ottenuti con SHIFT o con COMMODORE non corrispondono a quelli disegnati sulla parte frontale.

A questo punto e' assolutamente necessario provare piu' volte tutti i tasti descritti. Vi consigliamo di eseguire almeno i seguenti esercizi:

.1) Accendete il calcolatore e provate a premere uno dopo l'altro tutti i tasti osservando l'effetto sullo schermo. Poi proseguite tenendo premuto prima il tasto SHIFT e poi il tasto COMMODORE. In questo modo avrete provato tutti i caratteri del set grafico. Vedrete che se voi continuate a premere tasti lo schermo si riempie e quando siete arrivati in fondo viene attivato lo scrolling e voi potete proseguire. Se pero' ad un certo punto premete il tasto RETURN vedrete il messaggio SYNTAX ERROR, infatti le vostre scritte non sono linguaggio BASIC.

Se ora premete insieme i due tasti SHIFT e COMMODORE, senza avere prima ripulito lo schermo (usando SHIFT e CLR HOME), vedrete che buona parte dei caratteri dello schermo passa alla rappresentazione corrispondente nel set maiuscolo/minuscolo.

Ora potete pulire lo schermo, mettervi nella condizione del set maiuscolo/minuscolo, rifare la prova di tutti i caratteri della tastiera, usando i tasti da soli o insieme ai tasti SHIFT o COMMODORE.
Annotate i tasti grafici che non corrispondono ai disegni della tastiera.

Potete anche provare il tasto CTRL. Lo scopo di questo primo esercizio e' quello di rendervi familiari i diversi tasti in modo che una scarsa conoscenza della tastiera non vi impedisca di procedere nell'apprendimento delle possibilita' del vostro calcolatore.

.2) Provate a premere il tasto del "2" insieme a SHIFT; ottenete i doppi apici. Se scrivete dei caratteri li vedrete comparire come al solito, ma se premete dei tasti comando tipo CLR HOME, i due CRSR, RVS ON insieme a CTRL, RVS OFF insieme a CTRL, che normalmente non producono un carattere sul video ma una azione, vedrete apparire dei caratteri in campo inverso. Questo dipende dal fatto che avete aperto gli apici. Provate a premere SHIFT e CLR HOME, cioe' pulite lo schermo. Premendo il tasto CRSR/freccia basso-alto fate arrivare il cursore circa a meta' schermo. Ora scrivete:

PRINT " poi premete contemporaneamente SHIFT e CLR HOME e chiudete gli apici; otterrete sul video la parola PRINT e poi tra doppi apici un cuore in campo inverso; ora premete il tasto RETURN, vedrete che lo schermo si ripulisce ed il cursore va nell'angolo alto a sinistra. Questo significa che avete dato un comando di PRINT seguito tra apici dal comando SHIFT e CLR HOME e il comando e' stato eseguito per effetto dell'esecuzione della PRINT.

Provate con questo sistema della PRINT seguita da comandi tra apici anche tutti gli altri tasti comandi e ricordate l'effetto ottenuto.

.3) Ripulite ora lo schermo e scrivete il vostro nome, poi scendete alla riga sotto, premete insieme CTRL e RVS ON, scrivete nuovamente il vostro nome. Vedrete il vostro nome in campo inverso sotto a quello scritto in modo normale. Se ora scendete di nuovo, premete insieme CTRL e RVS OFF e poi scrivete ancora il nome esso sara' di nuovo scritto normalmente. Avete imparato ad usare i due tasti comandi RVS ON e RVS OFF. Provate ora ad usarli insieme a PRINT all'interno degli apici e ricordatevi quanto avete ottenuto.

.4) Provate ora a correggere qualcosa che avete scritto sullo schermo usando i due tasti CRSR per spostare il cursore al punto voluto, il tasto INST DEL per cancellare caratteri ed i due tasti SHIFT e INST DEL per creare spazi dove inserire nuovi caratteri.

.5) Ponetevi di fronte alla tastiera e chiedetevi a cosa serve ogni tasto; se sapete rispondere bene per ciascuno di essi potete andare avanti. In caso contrario rileggete tutto fino a questo punto. Non protestate, vi prego, ma vi accorgerete che e' assolutamente necessario conoscere bene la tastiera. Un po' di pazienza adesso vi evitera' inutili perdite di tempo piu' avanti.

2.2. LO SCHERMO

Accendete il vostro VIC ed osservate lo schermo. Vedete un bordo esterno azzurro (BORDO) che circonda un rettangolo bianco (SFONDO). Sullo sfondo bianco appaiono delle scritte blu ed il cursore lampeggiante e' blu. Questa e' la situazione dello schermo al momento dell'accensione. In seguito impareremo a modificare i colori. Quando voi lavorate vi potete muovere con il cursore solo entro il rettangolo interno.

Le dimensioni dello sfondo per il calcolatore sono di 23 righe di 22 colonne ciascuna, come appare dallo schema che segue.

```
                         COLONNE

     0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
     1
     2
     3
     4
     5
     6
     7
R    8
I    9
G   10
H   11
E   12
    13
    14
    15
    16
    17
    18
    19
    20
    21
    22
```

Questo corrisponde ad avere davanti a se' un foglio di carta a quadretti, avente 22 quadretti per riga (numerati da 0 a 21) e 23 righe (numerate da 0 a 22), cioe' in tutto 22 x 23 = 506 quadretti.

Lo schermo puo' contenere 506 caratteri scrivendone 22 per ogni riga e scrivendo 23 righe.

Provate a scrivere esattamente quanto segue:

```
10 REM PROVA NUMERO RIGHE E COLONNE
15 PRINT CHR$(147)
20 FOR K=1 TO 22
25 PRINT "Q";
30 NEXT K
40 FOR J=1 TO 21
45 PRINT "Q"
50 NEXT J
60 PRINT "Q";
70 GET A$
80 IF A$="" THEN 70
```

premendo il tasto RETURN alla fine di ogni riga. Poi scrivete:

RUN e premete RETURN.

Vedrete apparire sulla prima linea 22 caratteri Q maiuscoli e sulla prima colonna 23 caratteri Q maiuscoli. Avete ora scritto ed eseguito un programma in linguaggio BASIC. E' troppo presto per capire il significato di ogni linea di programma, ma abbiate fiducia e proseguite nello studio del VIC.

2.3. COLLEGAMENTO TRA SCHERMO E TASTIERA

Nei due precedenti paragrafi abbiamo parlato delle principali caratteristiche dello schermo e della tastiera, ora cercheremo di approfondire l'argomento. Sappiamo gia' che quando premiamo un tasto sulla tastiera, se questo produce un carattere stampabile, esso appare nella posizione dello schermo dove si trovava il cursore. Sappiamo anche manovrare il cursore per portarci dove vogliamo sullo schermo.

Abbiamo visto che lo schermo puo' contenere 506 caratteri. Possiamo chiederci dove va a finire un carattere quando azioniamo un tasto e vediamo apparire il carattere sullo schermo.

Non ne abbiamo ancora parlato, ma tutti sanno che i calcolatori elettronici hanno una memoria; essa e' un dispositivo dove stanno i programmi che noi scriviamo ed eseguiamo, i dati di calcolo, le scritte che compaiono sullo schermo. La memoria possiamo immaginarla come formata da una serie di piccoli contenitori (cassettini o caselline) posti uno di seguito all'altro. Ognuno di questi contenitori prende anche il nome di byte. Ogni byte contiene 8 bit (vedi paragrafi 6.2. e 6.3.).

Nella scritta che compare sul video al momento dell'accensione del VIC, nella seconda riga vediamo un numero (che puo' essere 3583 o altro a seconda delle dimensioni della memoria del calcolatore) seguito dalle parole BYTES FREE. Questo significa che possiamo disporre di quel numero di contenitori (BYTES) per lavorare con il VIC.

Ogni casellina e' contrassegnata da un numero; questo numero rappresenta l'indirizzo della casellina ed e' necessario per poter distinguere quella casellina da tutte le altre.

Le 506 posizioni dello schermo corrispondono esattamente a 506 BYTE nella memoria del calcolatore. L'indirizzo del primo byte e' 7680. Esso corrisponde al primo carattere della prima riga in alto a sinistra sul video. L'indirizzo del byte che corrisponde all'ultimo carattere in alto a destra nella prima riga e' 7701.

Potete ricavare facilmente gli indirizzi di tutti i bytes che costituiscono quella che chiameremo spesso la "MEMORIA DI SCHERMO" consultando lo schema che segue.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
7680	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7702	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7724	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7746	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7768	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7790	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7812	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7834	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7856	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7878	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7900	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7922	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7944	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7966	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7988	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8010	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8032	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8054	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8076	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8098	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8120	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8142	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8164	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

In esso lo schermo e' rappresentato da 506 quadrettini (punti nel disegno) disposti in 23 righe di 22 caratteri ciascuna. Sommando il numero scritto alla sinistra della riga con il numero scritto sopra la colonna alle quali appartiene la posizione voluta si ottiene l'indirizzo del byte corrispondente.

Fate ora questa prova:

. premete contemporaneamente i tasti RUN STOP e RESTORE; vedrete lo schermo pulito, la scritta READY sulla seconda riga e il cursore blu sotto la R;

. scrivete ora esattamente quanto segue ed usate il tasto RETURN alla fine della riga:

```
PRINT PEEK(8164)
```

vedrete apparire 32 sul video. Infatti la prima posizione a sinistra dell'ultima riga dello schermo e' pulita e contiene uno spazio che viene rappresentato nella memoria del VIC con

il numero 32. Avreste ottenuto ancora il numero 32 usando invece di 8164 l'indirizzo di un qualunque byte corrispondente all'ultima riga del video nella memoria di schermo, infatti l'ultima riga e' tutta pulita.
Il risultato ottenuto dipende dall'avere usato il comando PEEK che va a leggere il contenuto di un byte della memoria ed il comando PRINT che stampa sul video quello che e' stato letto.

. Scrivete ora, usando sempre RETURN alla fine:

```
PRINT PEEK(7702)
```

vedrete apparire il numero 18 sul video. Infatti 18 e' il codice corrispondente alla lettera R nel set grafico di caratteri, e 7702 corrisponde alla prima posizione della seconda riga, dove c'e' R. Se andate a consultare la tabella dell'Appendice B trovate proprio questi codici numerici in corrispondenza a spazio ed R nel primo set di caratteri.

Dopo questa prova avete cominciato a capire l'utilizzo da parte del VIC della memoria di schermo.

Abbiamo parlato di codici corrispondenti ai caratteri che appaiono sullo schermo. Facciamo riferimento all'Appendice A dove sono riportati il set dei caratteri grafici, il set maiuscolo/minuscolo e i codici dei caratteri. Si parla di codici ASCII, infatti la codifica dei caratteri nel VIC ha questo nome. I codici numerici variano da 0 a 255. Un byte puo' contenere un numero avente un valore compreso tra 0 e 255. Abbiamo quindi a disposizione 256 codici diversi, pero' non abbiamo a disposizione 256 caratteri stampabili diversi. Osservando la tabella dell'Appendice A vedrete che alcuni codici corrispondono a comandi e che a gruppi di codici diversi corrispondono gli stessi caratteri.

Proviamo ora i caratteri stampabili dei due set scrivendo un programma in BASIC. Eseguite esattamente quanto suggerito:

. scrivete NEW e poi premete RETURN;

. premete contemporaneamente RUN STOP e RESTORE;

. scrivete il programma che segue usando il tasto RETURN dopo ogni linea:

```
10 PRINT CHR$(147)
15 L=0
20 PRINT "CARATTERI STAMPABILI"
25 PRINT "  SET GRAFICO"
30 PRINT
```

```
 35 FOR K=32 TO 127
 40 PRINT K,CHR$(K)
 45 PRINT
 50 FOR J=1 TO 500:NEXTJ
 55 NEXTK
 60 FOR K=160 TO 255
 65 PRINT K, CHR$(K)
 70 PRINT
 75 FOR J=1 TO 500:NEXTJ
 80 NEXTK
 85 PRINT
 90 IF L=1 THEN STOP
 95 PRINT CHR$(147)
100 PRINT CHR$(14)
110 PRINT "CARATTERI STAMPABILI"
115 PRINT "SET MAIUSCOLO/MINUSCOLO"
120 L=1
130 GOTO 30
```

. scrivete RUN e poi premete RETURN. Vedrete apparire sullo schermo a sinistra, con uno scrolling abbastanza lento, il codice ASCII del carattere e a destra il carattere corrispondente, dapprima per il set grafico e poi per il set maiuscolo/minuscolo. Non spieghiamo ora il programma, ma ci limitiamo ad osservare che la funzione CHR$(K) produce il carattere corrispondente nella codifica ASCII al valore della variabile K. Inoltre ricordate che eseguendo il comando PRINT CHR$(14) si passa dal set grafico all'altro ottenendo lo stesso effetto di quando si premono contemporaneamente i tasti SHIFT e COMMODORE. Se invece si esegue il comando PRINT CHR$(142) si passa al set grafico. Il comando PRINT CHR$(147) permette di ottenere lo stesso risultato che si ha quando si premono contemporaneamente i tasti SHIFT e CLR HOME.

Abbiamo usato i codici ASCII da 32 a 127 e da 160 a 255 perche', come potete vedere dall'Appendice A, per questi valori si hanno caratteri stampabili.

Facciamo ora un altro piccolo esercizio:

. premete contemporaneamente RUN STOP e RESTORE;

. scrivete PRINT 97,CHR$(97) e poi RETURN;

. vedrete apparire sulla quarta riga del video a sinistra il numero 97 e un "picche" nella dodicesima colonna;

. scrivete PRINT PEEK(7747),PEEK(7748),PEEK(7757) e poi RETURN;

. vedrete apparire: 57, 55, 65; infatti abbiamo letto le posizioni corrispondenti al 9 ed al 7 di 97 ed al picche. I

numeri 57, 55 e 65 sono i codici che corrispondono rispettivamente ai caratteri 9, 7 e picche nella tabella della Appendice B relativa ai contenuti della memoria di schermo, quando si usa il comando PEEK.

A questo punto e' necessario spiegare le differenze tra le tabelle delle dua Appendici A e B.

Il linguaggio BASIC tratta i caratteri riportati nella Appendice A e nel primo set di caratteri il codice ASCII di picche e' 97. Noi giustamente abbiamo usato il comando BASIC PRINT 97, CHR$(97) per ottenere il picche. Solo che il VIC ha scritto il 97 e il picche nella memoria di schermo usando il suo codice interno che e' quello riportato nell'Appendice B ed e' leggermente diverso dal codice ASCII.
Vi sembrera' un po' complicato, ma fortunatamente queste trasformazioni di codice le fa il calcolatore da solo e noi, se vogliamo andare a vedere fino in fondo cosa succede, abbiamo le tabelle.

Per chi vuol capire meglio facciamo queste considerazioni:

. se osservate la tabella della Appendice A dei codici ASCII vedete che per ambedue i set di caratteri i gruppi di codici da 0 a 31 e da 128 a 159 non corrispondono a caratteri stampabili mentre tutti gli altri codici corrispondono a caratteri stampabili;

.confrontate la tabella della Appendice B con quella della Appendice A e vedrete che per la trasformazione dei codici valgono le seguenti regole, chiamando A il codice ASCII e D il DISPLAY CODE valgono le seguenti relazioni:

se 32 <= A <= 63 allora 32 <= D <= 63

se 64 <= A <= 95 allora 0 <= D <= 31 quindi D=A-64

se 96 <= A <= 127 allora 64 <= D <= 95 quindi D=A-32

se 160 <= A <= 191 allora 96 <= D <= 127 quindi D=A-64

2.4. IL COMANDO PRINT

Nei precedenti paragrafi abbiamo usato diverse volte il comando PRINT e oramai sappiamo che, in qualche modo, compare sul video quello che viene dopo questa parola chiave.

Riguardando gli esercizi svolti cominciamo a notare che la parola chiave PRINT e' stata usata in due modi diversi:
. facendola precedere da un numero;
. non facendola precedere da un numero.

Nel primo caso per far eseguire il comando dovevamo scrivere RUN e poi premere RETURN; nel secondo bastava premere RETURN.

Il VIC puo' eseguire le frasi del BASIC in due modi diversi:
. modo differito o indiretto;
. modo immediato o diretto.

Nel modo differito prima si scrive il programma linea per linea premendo RETURN alla fine di ogni linea e numerando le linee con numeri progressivi. Le linee di programma, oltre a comparire sul video vengono memorizzate e quando si scrive RUN e poi si preme RETURN il programma va in esecuzione e le PRINT lavorano.

Nel modo immediato si scrive la linea di comandi senza farla precedere da un numero ed essa va in esecuzione quando si preme RETURN e la PRINT lavora.

Quello che viene stampato sul video dipende da cosa c'e' dopo PRINT. Facciamo delle prove. E' sottinteso che prima di ogni nuova prova lo schermo deve essere pulito e questo si ottiene o premendo contemporaneamente RUN STOP e RESTORE o premendo contemporaneamente SHIFT e CLR HOME. Vedrete in seguito che queste due procedure danno solo apparentemente gli stessi risultati, ma ora per noi e' indifferente usare l'una o l'altra.

Scrivete:

PRINT 123,457 e poi premete RETURN

vedrete apparire sul video 123 a partire dalla seconda colonna e 457 in centro alla riga. Il calcolatore quando stampa un numero lo fa precedere da uno spazio o dal segno meno, se il numero e' negativo. Se contate le posizioni vedrete che 457 inizia sulla tredicesima colonna, infatti la "virgola" che separa i due numeri da stampare ha l'effetto di produrre uno spostamento alla dodicesima colonna, ma poi viene lasciato uno spazio prima del numero.

Scrivete ora:

```
PRINT -324,-1980
```

vedrete apparire:

```
-324         -1980
```

questo e' conforme a quanto detto sopra.

Scrivete ora:

NEW e poi premete RETURN

questo comando, dato in modo immediato, serve per cancellare dalla memoria del calcolatore un precedente programma, cioe' serve per ripulire la memoria del calcolatore, ma non quella di schermo. Per pulire lo schermo sapete gia' come fare. Ora scrivete:

```
10 PRINT 1678,4567
20 PRINT -567,-890
```

poi scrivete: RUN e poi premete RETURN. Il programma va in esecuzione e vedrete sullo schermo:

```
 1678         4567
-567         -890
```

i vostri numeri sono incolonnati a partire dalle cifre piu' significative. Vedremo in seguito come ottenere l'incolonnamento sulle cifre meno significative.

Scrivete ora:

```
10 PRINT 1678;4567
20 PRINT -567;-890
```

e poi fate girare il programma. Vedrete:

```
 1678  4567
-567 -890
```

Cosa e' successo? Abbiamo separato i numeri da stampare con il punto e virgola invece che con la virgola e il secondo numero e' venuto vicino al primo con due spazi di separazione nella prima stampa, mentre nella seconda stampa si ha un solo spazio di separazione e poi il segno.

La regola e' la seguente:

. il separatore virgola produce un salto alla zona di

stampa che inizia alla dodicesima colonna;

. il formato di stampa dei numeri e': uno spazio o il segno meno, le cifre del numero, uno spazio;

. il separatore punto e virgola fa stampare i numeri vicini rispettando il loro formato di stampa.

Scrivete ora:

```
PRINT "STO FACENDO PROVE DI STAMPA","BELLO"
```

e premete RETURN. Vedrete:

```
STO FACENDO PROVE DI S
TAMPA       BELLO
```

Vediamo di capire cosa e' successo. Dopo la PRINT abbiamo chiesto di stampare due stringhe di caratteri e le abbiamo scritte tra apici. Il carattere separatore e' la virgola e quindi deve produrre il salto alla dodicesima colonna; i caratteri della prima stringa sono piu' di 22 e quindi il calcolatore va a capo con quelli che non entrano nella prima riga, poi salta alla dodicesima colonna e scrive la seconda stringa.

Provate ora a scrivere ed eseguire:

```
PRINT "ALTRA PROVA","BELLO"
```

vedrete:

```
ALTRA PROVA
BELLO
```

infatti la prima stringa e' lunga 11 caratteri e quindi non e' rimasto nulla da saltare, allora il calcolatore salta alla nuova linea per la seconda stringa.

Se ora provate con:

```
PRINT "ALTRA PROV","BELLO"
```

vedrete:

```
ALTRA PROV BELLO
```

infatti essendo la prima stringa minore di 11 caratteri gli e' rimasto qualcosa da saltare ed ha scritto la seconda stringa a partire dalla dodicesima colonna.

Se ora provate a scrivere ed eseguire:

```
PRINT "ALTRA PROVA";"BELLO"
```

vedrete:

```
ALTRA PROVABELLO
```

cioe' le due stringhe attaccate senza spazi separatori.

La regola e':

. le stringhe (insiemi di caratteri) sono stampate come sono senza spazi prima e dopo, cioe' il formato di stampa delle stringhe non comporta aggiunte di spazi;

. se il separatore e' la virgola e la stringa e' al massimo di 10 caratteri si ha il salto alla dodicesima colonna;

. se il separatore e' il punto e virgola le stringhe vengono stampate attaccate.

Scrivete ora:

```
10 PRINT 23;45;-89;
20 PRINT 24;12;-9
30 PRINT "HO FATTO LA PROVA"
40 PRINT "DEL ; FINALE"
50 PRINT 23,17,-78,
60 PRINT 89,-56,78
70 PRINT "HO FATTO LA PROVA"
80 PRINT "DELLA , FINALE"
```

e fate girare il programma. Vedrete:

```
 23  45 -89  24  12 -9

HO FATTO LA PROVA
DEL ; FINALE
 23         17
-78         89
-56         78
HO FATTO LA PROVA
DELLA , FINALE
```

Dopo il -9 della prima riga e' andato a capo per lo spazio dopo il 9 (con il 9 sono 22 caratteri) e poi e' andato a capo un'altra volta perche' la riga di PRINT termina senza virgola o punto e virgola.

La regola e':

. il punto e virgola finale e la virgola finale non fanno andare a capo e il prossimo PRINT continua a scrivere sulla

stessa linea, mantenendo pero' valido il significato del separatore usato.

Negli esempi visti in questo paragrafo abbiamo stampato con PRINT dei dati costanti, cioe' dei numeri o delle stringhe. Avevamo pero' visto precedentemente che scrivendo:

PRINT CHR$(K)   dove K e' un numero prelevato dalla tabella dei codici ASCII della appen=dice A

si ottiene il corrispondente carattere stampabile. Inoltre avevamo visto che scrivendo:

PRINT PEEK(X)   dove X e' l'indirizzo di un byte del=la memoria di schermo

si ottiene la stampa del codice numerico contenuto nel byte e reso disponibile dalla funzione PEEK.

Ora nel prossimo paragrafo vedremo come si fa a introdurre nella memoria del VIC dati variabili. Vedremo poi come si possono stampare anche questi dati variabili.

Ora che avete capito il comando PRINT potete imparare a scriverlo in modo abbreviato. Se voi scrivete all'inizio della frase BASIC il punto interrogativo "?" il sistema lo interpreta come PRINT. Provate a ripetere qualche prova e abituatevi a questo risparmio di tempo.

2.5. LE VARIABILI E IL COMANDO INPUT

Lo scopo di questo paragrafo e' quello di imparare a introdurre dati, numeri o stringhe, nella memoria del VIC, usando i comandi BASIC.

Seguite le istruzioni che seguono:

scrivete NEW e poi premete RETURN

premete RUN STOP e RESTORE contemporaneamente

scrivete quanto segue premendo RETURN dopo ogni linea:

```
10 INPUT A
20 PRINT "HO INTRODOTTO: ";A
```

scrivete RUN e poi premete RETURN

vedrete apparire un punto interrogativo sul video. A questo punto interrogativo rispondete scrivendo un numero,

per esempio 546, e poi premete RETURN.

Vedrete apparire sul video:

HO INTRODOTTO: 546

Infatti voi avete scritto un programma che chiede all'utente di scrivere un numero, questo numero viene memorizzato in un contenitore denominato A e poi viene stampata la frase HO INTRODOTTO: seguita dal contenuto del contenitore A.

Vi preghiamo di ricordare che rispondendo ad una richiesta di INPUT si deve usare il tasto RETURN per fare accettare i dati dal calcolatore. Se ora voi scrivete ancora RUN e poi premete RETURN, vedrete apparire ancora il punto interrogativo. Se questa volta rispondete con -418, vedrete apparire:

HO INTRODOTTO: -418

ed il numero -418 e' andato a sostituire nel contenitore A il suo precedente contenuto.

Che cosa e' A? A e' il nome di un contenitore che viene piu' propriamente chiamato variabile. Come abbiamo visto nell'esempio il contenuto di A varia e quindi il nome e' appropriato.

Cosa ne e' di A nella memoria del calcolatore? Alla variabile A viene assegnato un angolino, uno spazio nella memoria del VIC, e quando le frasi di un programma chiamano A si va a mettere qualcosa o a prendere qualcosa in quello spazio. Nel nostro esempio la linea di programma 10 va a mettere qualcosa in A, e precisamente quello che legge dalla tastiera quando l'utente risponde al punto interrogativo con un numero e preme RETURN. La linea di programma 20 va a prendere qualcosa dalla variabile A e infatti stampa il suo contenuto sul video. Pero' la frase PRINT non danneggia il contenuto di A, lo lascia invariato. Se invece facciamo girare un'altra volta il programma il contenuto di A viene modificato; infatti la linea 10 va di nuovo a mettere qualcosa in A sostituendolo al precedente contenuto.

E se vogliamo memorizzare una frase? Provate a scrivere quanto segue, dopo aver cancellato il programma precedente e ripulito il video:

```
10 INPUT A$
20 PRINT "HO LETTO:"
30 PRINT A$
```

ed ora fate girare il programma. Vedrete di nuovo il punto interrogativo, ma questa volta rispondete con una frase, per esempio: CIAO BELLO e non dimenticate l'indispensabile RETURN. Vedrete apparire:

```
HO LETTO:
CIAO BELLO
```

In questo caso la nostra variabile si chiama A$ e contiene una frase. Se volete potete far girare ancora il programma e rispondere con altre frasi al punto interrogativo.

Avrete notato che nel primo programma volendo introdurre numeri abbiamo chiamato A la variabile, mentre nel secondo volendo introdurre frasi abbiamo chiamato A$ la variabile. Provate a far girare il primo programma e a rispondere al punto interrogativo con una frase; vedrete apparire sul video REDO FROM START e poi di nuovo il punto interrogativo. Questo significa che non puo' accettare una frase con la variabile A, mentre l'accetta con la variabile A$. Quale e' la differenza nei nomi delle variabili? Il simbolo $ finale.

Abbiamo imparato che l'utente puo' inventare dei nomi per le variabili; la regola per la formazione dei nomi e' la seguente:

. i nomi delle variabili iniziano con una lettera dell'alfabeto;

. i nomi possono essere formati da una lettera sola o da una lettera seguita da un'altra lettera o da una cifra;

. se le variabili si riferiscono a frasi (stringhe di caratteri alfabetici, numerici o speciali) il nome della variabile deve terminare con il carattere $.

Esempi di nomi per variabili numeriche:

A, CV, K1, P9, XX, ecc.

Esempi di nomi per variabili stringa:

A$, CV$, K1$, P9$, XX$, ecc.

Procediamo ora nello studio del comando INPUT. Tenete presente che questo comando non puo' essere eseguito in modo immediato. Provate ed avrete una segnalazione di errore.

Proviamo a leggere piu' variabili con un solo comando INPUT. Scrivete e fate girare il programma che segue:

```
10 INPUT A,B,C
20 PRINT A;B;C
```

vedrete apparire il solito punto interrogativo, ma la frase INPUT chiede di riempire le tre variabili A, B e C. Potete rispondere, per esempio:

```
?27,-123,97
```

e vedrete apparire:

```
 27 -123  97
```

Se rispondete solo, per esempio 45 e poi premete RETURN, vedrete apparire due punti interrogativi vicini; essi servono per avvisarvi che non avete soddisfatto pienamente le richieste della frase INPUT e che dovete scrivere altri numeri. Avrete notato che il calcolatore considera terminato il primo numero quando voi usate la virgola. La virgola viene quindi usata come separatore durante la lettura dei numeri.

La frase di lettura dati dalla tastiera puo' essere ulteriormente migliorata se si scrive cosi':

```
10 INPUT "SCRIVI 3 NUMERI";A,B,C
20 PRINT A;B;C
```

facendo girare questo programma si ottiene:

```
SCRIVI 3 NUMERI?
```

e voi rispondete con i 3 numeri separati da virgola, ma sapete che dovete introdurre 3 numeri. Notate che se fate seguire alla parola chiave INPUT una frase tra apici, dopo ci vuole il punto e virgola prima delle variabili da leggere, separate tra loro da virgole.
La frase tra apici non puo' superare i 20 caratteri.

Vediamo ora di leggere due variabili stringa. Scrivete e fate girare questo programma:

```
10 INPUT"SCRIVI 2 FRASI";A$,B$
20 PRINT A$
30 PRINT B$
```

vedrete apparire:

SCRIVI 2 FRASI? e voi potete rispondere con 2 frasi separate da virgola, pero' non potete usare la virgola come parte della frase.

Se desiderate introdurre delle frasi che comprendano anche il carattere virgola dovete includerle tra gli apici. Nella situazione di prima potete rispondere:

"PIOVE, PECCATO",PIPPO

ed il calcolatore stampera':

PIOVE, PECCATO
PIPPO

il contenuto di A$ e': PIOVE, PECCATO e quello di B$: PIPPO.

Se alla richiesta di INPUT, cioe' al punto interrogativo rispondete solo con il tasto RETURN, il calcolatore accetta la stringa nulla se la variabile si riferisce ad una stringa e zero se la variabile si riferisce ad un numero. Per uscire dalla richiesta di INPUT potete premere contemporaneamente RUN/STOP e RESTORE oppure SHIFT destro e RUN/STOP.
A questo punto vi consigliamo di sbizzarrirvi con varie prove di piccoli programmi che usano i comandi INPUT e PRINT, i dati costanti e le variabili.

2.6. ANCORA VARIABILI

Nel paragrafo precedente abbiamo introdotto il concetto di variabile ed abbiamo fatto delle prove imparando a distinguere le variabili numeriche dalle variabili stringa. Ora dobbiamo procedere su questo importante argomento e introdurre la distinzione tra variabili che contengono numeri interi e variabili che contengono numeri decimali.

Scrivete ed eseguite il seguente programma:

```
10 INPUT "SCRIVI NUM. INTERO";A%
20 PRINT A%
```

e rispondete alla richiesta di INPUT con un numero intero; lo vedrete regolarmente stampato dalla PRINT. Se ora fate girare nuovamente il programma e rispondete alla richiesta di INPUT con un numero decimale; lo vedrete stampato troncato della parte decimale, ma senza segnalazione di errore.

Abbiamo sperimentato che facendo terminare il nome di una variabile con il suffisso "%" essa accetta solo numeri interi, troncando eventualmente i decimali.

Dobbiamo quindi aggiungere alle regole sui nomi delle variabili del precedente paragrafo anche questa:

. se il nome di una variabile termina con il carattere %, la variabile contiene numeri interi.

Naturalmente si possono usare le variabili numeriche che non terminano con % per memorizzare anche numeri interi, ma nella grammatica del BASIC queste variabili sono definite come decimali ed occupano piu' spazio in memoria delle variabili definite come intere.

Vediamo ora quale grandezza ammette il VIC per i contenuti delle variabili.

Le VARIABILI INTERE possono contenere numeri compresi tra -32768 e +32767.

Le VARIABILI DECIMALI, che vengono anche chiamate Floating Point, possono contenere numeri in valore assoluto compresi tra 2.93873588 moltiplicato 10 elevato a -39 e 1.70141183 moltiplicato 10 elevato a +38.

Le variabili stringa possono contenere fino a 255 caratteri, pero' in fase di INPUT si possono introdurre al massimo 88 caratteri; le stringhe poi possono essere concatenate tra loro e formare una stringa piu' lunga, come vedremo piu' avanti.

Quando i numeri vengono stampati si puo' avere qualche sorpresa per i numeri decimali. Ricordiamo che viene usato il punto decimale invece della virgola. Il VIC stampa un numero decimale con il punto decimale fino a quando il numero e' compreso in valore assoluto tra 0.01 e 999999999. Se il numero cade furi da questo intervallo esso viene stampato in notazione scientifica e cioe' :
- . il segno;
- . la prima cifra significativa;
- . seguita dal punto decimale;
- . seguita dalle altre cifre (fino ad 8 cifre);
- . seguita dalla lettera E (che significa molti= plicato 10 elevato a);
- . seguito da + o - l'esponente a cui elevare 10.

Cosi':
5.34567898E+12.

La precisione dei numeri e' di circa 9 cifre. Si possono avere alcune sorprese nei calcoli, dovute al modo come il calcolatore lavora (in aritmetica binaria).
Provate a scrivere:

?1.70141184E+38

premendo RETURN ottenete:

```
?OVERFLOW ERROR
READY.
```

e questo significa che avete scritto un numero troppo grande ed avete superato la capacita' del calcolatore.

Se invece scrivete:

```
?2,93873587E-39
```

premendo RETURN ottenete:

```
0
READY.
```

e questo significa che avete scritto un numero troppo piccolo ed il calcolatore lo arrotonda a zero.

Se date in INPUT un numero fuori dall'intervallo consentito esso viene arrotondato con le solite regole matematiche: aggiunge 1 se si trascura una cifra superiore a 5.

Dobbiamo ora aggiungere qualcosa circa la formazione dei nomi delle variabili. Infatti i nomi delle variabili, inventati dal programmatore, possono anche essere piu' lunghi di due caratteri, ma i caratteri, oltre il secondo e prima dell'eventuale suffisso % o $, non vengono considerati. Bisogna quindi stare attenti perche' le variabili PINO e PIENO per il calcolatore sono la stessa variabile.

Si deve inoltre fare attenzione nell'inventare i nomi delle variabili a non usare le parole chiave del linguaggio BASIC, facendo attenzione soprattutto a quelle di 2 caratteri. Per comodita' riportiamo l'elenco delle parole riservate del BASIC.

Si raccomanda comunque di non usare nomi lunghi per le variabili onde evitare di sprecare spazio in memoria. Il vantaggio dei nomi lunghi e' solo che sono piu' significativi (mnemonici) di quelli corti e rendono piu' leggibili i programmi quando essi vengono listati.

TABELLA PAROLE RISERVATE

ABS	GET	ON	SPC
AND	GET#	OPEN	SQR
ASC	GOSUB	OR	ST
ATN	GOTO	PEEK	STEP
CHR$	IF	POKE	STOP
CLOSE	INPUT	POS	STR$
CLR	INPUT#	PRINT	SYS
CMD	INT	PRINT#	TAB
CONT	LEFT$	READ	TAN
COS	LEN	REM	THEN
DATA	LET	RESTORE	TI
DEF	LIST	RETURN	TI$
DIM	LOAD	RIGHT$	TO
END	LOG	RND	USR
EXP	MID$	RUN	VAL
FN	NEW	SAVE	VERIFY
FOR	NEXT	SGN	WAIT
FRE	NOT	SIN	

(GOTO puo' essere scritto anche GO TO)

2.7. FACCIAMO DEI CALCOLI

Proviamo ora ad usare il VIC per fare dei calcoli. Scrivete usando il calcolatore in modo immediato:

```
?27*18
```

otterrete:

```
486
```

infatti il calcolatore ha eseguito il prodotto di 27 per 18 ed ha stampato il risultato.

Provate ora:

```
?1324/(54*2)
```

otterrete: 12.2592593

se ora provate:

```
?1324/54*2
```

otterrete: 49.037037

infatti nel primo calcolo, a causa delle parentesi prima e' stato calcolato il prodotto di 54 per 2 e poi e' stato

diviso 1324 per il risultato della moltiplicazione; nel secondo invece prima e' stato moltiplicato 1324 per 54 e poi il risultato e' stato diviso per 2.

Vediamo di dedurre alcune regole:

. le operazioni sono svolte da sinistra a destra, ma hanno la precedenza quelle racchiuse in parentesi. Sono cioe' mantenute le solite regole della matematica;

. l'asterisco si usa per il simbolo di moltiplicazione;

. la barra trasversale a destra si usa per il simbolo di divisione.

Il VIC calcola quindi anche complicate espressioni con l'uso degli operatori aritmetici e delle parentesi. Inoltre si possono usare gli operatori relazionali e gli operatori logici o booleani.

Le espressioni vengono calcolate muovendosi da sinistra verso destra, ma dando alle diverse operazioni una precedenza che chiamiamo priorita'. Vengono eseguite prima le operazioni che hanno la priorita' piu' alta considerando anche la priorita' delle parentesi.

Gli operatori relazionali possono entrare nelle espressioni aritmetiche e alla relazione viene sostituito il valore -1 se essa e' verificata ed il valore 0 se essa non e' verificata. Esempio:

```
?27*(23<6)   da' come risultato 0
```

infatti 23<6 non e' una relazione vera e quindi gli viene sostituito 0.

Invece:

```
?27*(6<23)   da' come risultato -27
```

infatti 6<23 e' una relazione vera e quindi gli viene sostituito -1.

Riepiloghiamo quindi la regola:

. se una espressione relazionale e/o logica e' vera essa vale -1;

. se una espressione relazionale e/o logica e' falsa essa vale 0.

Per quanto riguarda gli operatori logici ne parleremo piu' diffusamente in seguito.

Riportiamo uno schema di tutti gli operatori segnalando la loro priorita' e le modalita' operative.

TIPO	PRIORITA'	OPERATORE	COMMENTI
	9	( )	parentesi che racchiu= dono espressioni
ARITMETICI	8	freccia su	elevamento a potenza
	7	-	segno numeri negativi
	6	*	moltiplicazione
	6	/	divisione
	5	+	addizione
	5	-	sottrazione
RELAZIONALI	4	=	uguale
	4	<>	non uguale
	4		minore
	4		maggiore
	4	<= o =<	minore o uguale
	4	>= o =>	maggiore o uguale
LOGICI	3	NOT	complemento logico
	2	AND	prodotto logico
	1	OR	somma logica

2.8. ESTRAIAMO DEI NUMERI A CASO

Provate a scrivere:

```
?RND(1);RND(0);RND(-1)
```

vedrete comparire sullo schermo 3 numeri quasi-a-caso. Essi potranno essere scritti o in notazione decimale o in

notazione esponenziale.

La funzione RND fornisce un numero compreso tra 0 e 1. Il sistema usa una formula matematica per calcolare i numeri a caso, da cui il "quasi-a-caso" usato prima. Questa formula genera una sequenza di numeri a caso partendo da un valore iniziale, detto seme (seed). La funzione si scrive RND(X), dove X puo' essere un numero positivo, negativo o nullo. Il numero X influisce sulla sequenza dei numeri generati in questo modo:

. se X=0 la sequenza inizia prendendo come origine il contenuto del contatore dei fotogrammi dello schermo;

. se X>0 la sequenza prosegue e il valore del numero non influisce sul numero generato;

. se X<0 viene preso come origine della sequenza il numero X e quindi per ogni X si ottiene un numero a caso diverso, ma sempre lo stesso per lo stesso X. Cioe' se volete che un vostro programma tratti sempre la stessa sequenza di numeri a caso dovete usare lo stesso numero X negativo con la funzione RND.

Potete fare questa prova: spegnete il VIC, riaccendetelo dopo qualche secondo, scrivete ed eseguite in modo immediato:

```
?RND(5)
```

prendete nota del numero ottenuto. Ripetete di nuovo la prova usando un numero diverso da 5 ma positivo, vedrete che il numero ottenuto e' sempre lo stesso. Questo significa che il calcolatore appena acceso si trova nelle stesse condizioni e non e' ancora stato generato alcun numero a caso, per cui per qualunque X positivo viene lo stesso numero random, dato che la partenza e' la stessa.

Se ripetete questa prova scrivendo invece RND(0) vedrete ogni volta un numero diverso.

Provate ad usare il programma che segue:

```
10 REM NUMERI RANDOM
20 INPUT"ARGOMENTO RND ";X
30 FOR K=1TO 10
40 PRINT RND(X)
50 NEXTK
60 PRINT
70 GOTO 20
```

esso chiede un numero X come argomento per la funzione RND e

scrive una sequenza di 10 numeri a caso, dopo va a capo e ritorna all'inizio. In tale modo potete ottenere diverse sequenze e studiare l'effetto dei diversi argomenti X. Se volete interrompere il programma mentre gira premete RUN STOP; se volete interrompere il programma mentre e' in attesa di INPUT premete RUN STOP e RESTORE insieme.

Provate ora questo altro programma, con il quale potete simulare il lancio di due dadi:

```
10 REM LANCIO DADI
20 D1%=RND(1)*6+1
30 D2%=RND(1)*6+1
35 PRINT
40 PRINT"PRIMO GIOCATORE"
50 PRINT"D1=";D1%,"D2=";D2%
55 PRINT
60 PRINT"SECONDO GIOCATORE"
70 D1%=RND(1)*6+1
80 D"%=RND(1)*6+1
90 PRINT"D1=";D1%,"D2=";D2%
95 FOR T=1 TO 2000: NEXTT
96 GOTO 20
```

vedrete apparire con continuita' il lancio dei dadi del primo e del secondo giocatore. Nelle linee 20, 30, 70 e 80 si ottiene con la RND un numero minore di 1, si moltiplica per 6 e si aggiunge 1, ottenendo cosi' un numero compreso tra 1 e 6; tale numero e' decimale, ma viene assegnato ad una variabile che termina con il carattere % e quindi reso intero troncando i decimali. Se volete interrompere il programma premete STOP RUN.

2.9. IL CAMPO INVERSO

Dopo aver ripulito lo schermo, scrivete e provate questo semplicissimo programma:

```
10 PRINT " <SHIFT e Q> ";
20 GOTO 10
```

vedrete delle palline blu in campo bianco e, per effetto dello scrolling, vi sembreranno in movimento. Abbiamo scritto <SHIFT e Q> entro gli apici e compresi tra minore e maggiore per indicare i tasti da premere contemporaneamente e non fare uso della funzione CHR$.

Fermate il programma con RUN STOP, fate la modifica che segue e riprovate il programma:

```
10 PRINT " <RVS ON><SHIFT eQ><RVS OFF> ";
```

vedrete la pallina di prima in campo inverso con lo stesso effetto di movimento precedente. Abbiamo scritto <RVS ON> per indicare il carattere corrispondente al tasto comando che si ottiene premendo CTRL e 9; e <RVS OFF> per indicare il carattere corrispondente ai tasti CTRL e 0. In questa PRINT si mantiene il campo inverso solo per stampare la pallina, mentre gli spazi vengono stampati in modo normale. RVS ON mette in campo inverso, RVS OFF fa tornare al modo diretto.

Se ora provate a far girare il programma con la modifica che segue:

```
10 PRINT "  <RVS ON><SHIFT e Q> ";
```

vedrete tutto lo schermo blu con delle palline bianche ed il solito movimento dovuto allo scrolling. Infatti nella print viene dato solo il comando RVS ON e quindi tutto, compresi gli spazi, viene stampato in campo inverso.

Potete scrivere e provare programmi usando i caratteri e gli spazi in campo inverso e vedrete che potete ottenere facilmente interessanti risultati grafici.

2.10. IL CONTROLLO DEL TEMPO

Il sistema controlla il passare del tempo aggiungendo 1 alla variabile TI ogni sessantesimo di secondo ed aggiornando di conseguenza anche TI$.

Al momento dell'accensione del VIC TI contiene 0 e TI$ contiene "000000". L'utente puo' caricare in TI$ l'ora giusta nella forma "HHMMSS", cioe' 2 caratteri per l'ora, 2 per i minuti e 2 per i secondi. Il sistema aggiorna di conseguenza TI e l'utente puo' leggere TI$ per sapere che ore sono e confrontare con il suo orologio.
Se lasciate acceso il VIC per piu' di un giorno a mezzanotte l'orologio viene azzerato.

Nei programmi si puo' controllare la durata di alcune operazioni memorizzando in una variabile il valore di TI o di TI$ all'inizio e poi memorizzandolo in un'altra variabile alla fine e facendo il confronto. Si puo' anche sospendere il programma per un certo tempo controllando quando TI si incrementa di un certo intervallo.

Potete fare delle prove scrivendo al momento dell'accensione del VIC:

```
TI$="HHMMSS"
```

e poi dopo un po' scrivere:

```
PRINT TI$
```

e controllare l'ora.

Ci sono comunque altri modi per controllare il passare del tempo. Uno e' quello di creare con il comando GET un ciclo di attesa nel programma scrivendo cosi':

```
10 GET A$: IF A$="" THEN 10
```

il programma prosegue quando si preme un qualunque tasto.

Un altro modo e' quello di usare il comando BASIC WAIT, ma questo modo lo raccomandiamo solo agli esperti di aritmetica binaria.

Un altro modo e' quello di usare un ciclo FOR a vuoto, usando una costante piu' o meno grande, cosi':

```
20 FOR K=1 TO 1000:NEXTK
```

in questo caso il programma prosegue senza l'intervento dell'operatore quando K e' arrivato a 1000. Potreste provare a scrivere cosi:

```
10 A=TI
20 FOR K=1 TO 1000:NEXTK
30 B=TI
40 PRINT A;B;B-A
```

cosi' potete vedere quanti sessantesimi di secondi dura il ciclo FOR con la costante 1000 come valore finale e partendo da 1.

2.11. LE VARIABILI CON INDICE

Supponiamo di voler leggere dalla tastiera 20 numeri decimali e di volerli memorizzare nella memoria del VIC per poterli elaborare in vari modi. Una soluzione e' quella di inventare 20 nomi di variabili diversi, come per esempio:

A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, B0, B1, B2, B3, B4, B5, B6, B7, B8, B9

e poi di scrivere 20 linee di programma che fanno INPUT di una di queste variabile per volta. In realta' e' un po' noioso e lungo da scrivere.

Un'altra soluzione e' quella di chiamare A tutte le 20 variabili e di distinguerle assegnando loro un indice numerico che vada da 1 a 20. Cosi':

A(1), A(2), A(3), ..., A(19), A(20).

Fortunatamente il BASIC ci consente di comportarci cosi'. Esiste una istruzione per dire al programma che si vuole lavorare con una variabile con indice, assegnarle un nome e le dimensioni, cioe quante variabili ci sono nel gruppo. Si scrive:

```
10 DIM A(20)
```

ed il programma accetta variabili scritte in questo modo:

A(K) oppure A(7) o qualunque altro numero tra parentesi purche' compreso tra 0 e 20, infatti il numero 20 e' stato usato per dimensionare la variabile.

Naturalmente anche K non deve mai superare 20 in questo caso.

Vediamo ora un programma che legge e poi stampa, 5 per riga i nostri 20 numeri:

```
 10 DIM A(20)
 20 FOR K=1 TO 20
 30 INPUT A(K)
 40 NEXTK
 50 FOR I=1 TO 4
 60 FOR K=1 TO 5
 70 PRINT A((I-1)*5+K);
 80 NEXT K
 90 PRINT
100 NEXTI
```

alla linea 10 viene definita la variabile A con indice e formata da 20 elementi, anzi da 21 perche' in BASIC gli indici cominciano da 0. Alle linee da 20 a 40 vengono letti i 20 numeri e memorizzati. Alla linea 50 viene iniziato un ciclo per stampare 4 righe. Alla linea 60 viene iniziato un ciclo interno al precedente per stampare 5 variabili per riga con indici crescenti da 1 a 20. In 80 si chiude il ciclo di K. In 90 viene data una PRINT a vuoto per andare a capo. In 100 si chiude il ciclo di I.

Vediamo la regola:

. la frase DIM dimensiona le variabili con indice;

. la parola chiave puo' essere seguita da piu'

dimensionamenti, cosi':

```
DIM A(5),B(6),C$(65)
```

. si possono avere variabili a piu' dimensioni, cosi':

```
DIM F(5,7),F$(3,4)
```

e in questo caso le variabili F e F$ hanno due dimensioni ciascuna e il numero di elementi e':

```
per F  (5+1)*(7+1)=48
per F$ (3+1)*(4+1)=20
```

. se si dimentica o non si vuole usare la frase DIM e si citano variabili con indice nel programma, il sistema attribuisce implicitamente il valore 10 ad ogni dimensione e quindi si hanno 11 elementi per ogni dimensione.

Le variabili con indice hanno un largo uso in programmazione, ma non e' un concetto facile da acquisire, di norma, le prime volte, viene confuso l'indice con il contenuto. Si tratta di fare un po' di esercizio.

Una variabile con indice a due dimensioni si presta molto bene a rappresentare tabelle, in questo caso il primo numero tra parentesi rappresenta il numero delle righe della tabella e il secondo il numero delle colonne.

CAPITOLO 3

IL COLORE

3.1. I COLORI DELLO SCHERMO

Quando accendete il VIC vedete lo schermo bianco con il bordo azzurro ed il cursore e le scritte blu. Sono quindi presenti 3 colori: uno per il bordo, uno per lo sfondo e uno per il cursore.

Una posizione di memoria del VIC controlla la combinazione di colori del video; e' il byte di indirizzo 36879.

Provate a scrivere ed eseguire il seguente programma, dopo il solito NEW e RETURN per ripulire la memoria e SHIFT e CLR HOME per ripulire il video:

```
10 REM PROVA COLORI
15 REM SFONDO SCHERMO
16 P$=CHR$(147)
17 CD$=CHR$(29)
18 CG$=CHR$(17)
20 FOR K=0 TO 255
30 POKE 36879,K
35 PRINT P$;CD$;CD$;CD$;CG$;CG$;CG$;CG$;
36 PRINT "POKE 36879,";K
40 FOR T=1 TO 1000: NEXTT
50 NEXTK
```

vedrete cambiare continuamente la combinazione dei colori sfondo schermo e per ogni combinazione apparira' in centro allo schermo nella parte alta la POKE necessaria per ottenere quella combinazione. Questo programma e' abbastanza complicato, vediamo cosa fa ogni linea.

Le linee 10 e 15 sono dei commenti ed iniziano con la parola chiave REM; servono per illustrare il contenuto del programma. Si possono mettere tutti i commenti desiderati in un programma, ma si occupa spazio di memoria.

La linea 16 serve per mettere nella stringa P$ il carattere di controllo SHIFT e CLR HOME.
La linea 17 serve per mettere nella stringa CD$ il carattere di controllo CRSR a destra.
La linea 18 serve per mettere nella stringa CG$ il carattere di controllo CRSR in giu'.

La linea 20 posiziona nella variabile K il valore iniziale 0 e fa iniziare un ciclo di programma che termina al NEXTK della linea 50; ogni volta che viene eseguita la linea 50 la variabile K viene incrementata di 1 e il programma ritorna alla linea 30.Il ciclo termina quando K ha raggiunto l'ultimo valore indicato nella linea 20 dopo la parola chiave TO, cioe' 255 ed ha lavorato anche con questo valore.

La linea 30 scrive con la POKE nel byte di controllo del colore, di indirizzo 36879 il numero contenuto al momento nella variabile K; questo numero corrisponde ad una delle 256 possibili combinazioni di colore.

La linea 35 fa una PRINT dei caratteri di controllo stampando le stringhe:

P$ che produce SHIFT e CLR HOME,
CD$ che fa spostare il cursore a destra di uno spazio,
CG$ che fa spostare il cursore in giu' di una linea,

e non va a capo.

La linea 36 scrive sullo schermo il comando POKE che e' stato dato per ottenere la combinazione di colori che appare, mettendo in evidenza il contenuto di K usato.

La linea 40 produce un ciclo di ritardo governato dal numero 1000 per consentire all'utente di leggere il numero della combinazione di colori usata.

Potete annotare le combinazioni che preferite ed usarle quando scrivete i vostri programmi.

Se non riuscite a studiare bene i colori perche' cambiano troppo rapidamente, basta aumentare nella linea 40 il numero 1000 fino ad ottenere il ritardo che vi piace.

Avrete osservato che per alcune combinazioni appare la scritta blu sullo sfondo, mentre in altre la scritta appare in bianco su una striscia blu, cioe' appare in campo inverso.

Nel VIC sono possibili 16 colori diversi per lo sfondo e 8 colori diversi per il bordo. Se moltiplichiamo 16 per 8 otteniamo 128. Questo e' il numero delle possibili combinazioni di colore sfondo/bordo. Per 128 combinazioni di colore si ha il cursore blu e le scritte appaiono in blu; per altre 128 combinazioni, che sono le medesime di prima, si ha il cursore modificato in modo che le scritte appaiono in campo inverso. Si hanno in tutto 256 combinazioni diverse tenendo conto anche dello stato del cursore e quindi del

modo come appaiono le linee di testo.

Per distinguere il colore del bordo si ha un numero che va da 0 a 7, mentre per distinguere il colore dello sfondo si ha un numero che va da 0 a 15. Nella tabella che segue sono riportati i numeri distintivi dei colori ed i loro significati.

NUMERI DISTINTIVI DEI COLORI

COLORE	NUM. PER SCHERMO S	NUM. PER BORDO B
NERO	0	0
BIANCO	1	1
ROSSO	2	2
AZZURRO	3	3
VIOLA	4	4
VERDE	5	5
BLU	6	6
GIALLO	7	7
ARANCIONE	8	
GIALLO/ARANCIO	9	
ROSA	10	
CELESTE	11	
VIOLA CHIARO	12	
VERDE CHIARO	13	
BLU INTERMEDIO	14	
GIALLO CHIARO	15	

Il numero da scrivere con la POKE nel byte di indirizzo 36879 si puo' calcolare con la seguente formula:

K = S * 16 + 8 + B

e si ottengono le 128 combinazioni di colore sfondo/bordo con il testo in campo diretto. Se non si aggiunge 8, cioe' si scrive:

K = S * 16 + B

si ottengono le 128 combinazioni di colore sfondo/bordo con il testo in campo inverso.

3.2. IL COLORE DEL TESTO

Nel precedente paragrafo abbiamo imparato a modificare i colori dello sfondo e del bordo. Ora dobbiamo imparare a modificare il colore del testo.

Nella parte in alto a destra della tastiera ci sono i

tasti da 1 a 8 che recano sulla parte frontale le abbreviazioni dei nomi in inglese dei primi 8 colori. Se noi premiamo contemporaneamente il tasto CTRL ed uno di questi tasti colore otteniamo il cambio del colore del cursore e i caratteri battuti dopo appaiono nel nuovo colore. Facendo queste prove potrete avere delle sorprese, infatti se ponete il cursore nel colore dello sfondo non vedete apparire i caratteri perche' essi sono dello stesso colore.

Si puo' ottenere il cambio del colore del testo anche utilizzando in comandi PRINT come dati i caratteri di controllo del colore; infatti:

```
CHR$(5)   corrisponde al bianco   e ai tasti CTRL e 2
CHR$(28)       "        " rosso    "  "   "      "  " 3
CHR$(30)       "        " verde    "  "   "      "  " 6
CHR$(31)       "        " blu      "  "   "      "  " 7
CHR$(144)      "        " nero     "  "   "      "  " 1
CHR$(156)      "        " viola    "  "   "      "  " 5
CHR$(158)      "        " giallo   "  "   "      "  " 8
CHR$(159)      "        " azzurro  "  "   "      "  " 4
```

Abbiamo quindi la possibilita' di scrivere il testo in 8 colori diversi.

Vediamo come primo esempio il semplice programma che segue, con il quale, dopo aver rimesso il VIC nelle condizioni iniziali (come con RUN STOP e RESTORE) di bordo azzurro e sfondo bianco, e questo si ottiene facendo POKE 36879,27, facciamo apparire 8 palline colorate, prima in campo diretto e poi in campo inverso.

```
10 REM PROVA COLORI
20 POKE 36879,27
30 PRINT CHR$(147)
40 C$=CHR$(144)+CHR$(5)+CHR$(28)+CHR$(159)
50 C$=C$+CHR$(156)+CHR$(30)+CHR$(31)+CHR$(158)
60 FOR K=1 TO 8
70 PRINT MID$(C$,K,1);CHR$(113);
80 NEXTK
85 PRINT CHR$(18);
87 FOR K=1 TO 8
89 PRINT MID$(C$,K,1);CHR$(113);
90 NEXTK
```

Vedrete che la seconda e la decima pallina mancano, infatti sono bianco su bianco e non si vedono.

La linea 20 pone il colore nelle condizioni iniziali di accensione del calcolatore. La linea 30 pulisce lo schermo e corrisponde ai tasti SHIFT e CLR HOME. La stringa C$ viene fabbricata alle linee 40 e 50 prendendo i caratteri che

corrispondono ordinatamente agli 8 colori nella stessa sequenza dei tasti di controllo del colore; viene messo in C$ il primo carattere e poi vengono "sommati" gli altri. Nelle linee da 60 a 80 c'e' il ciclo che stampa le prime 8 palline prelevando il carattere del colore dalla stringa C$ con la funzione MID$; la pallina e' ottenuta con CHR$(113). La linea 85 stampa il carattere di controllo per passare in campo inverso. Le linee da 87 a 90 rifanno il ciclo di stampa delle palline colorate.

3.3. LE POKE PER IL COLORE DEL TESTO

Oramai sappiamo che il comando POKE scrive qualcosa in un byte, lo abbiamo sperimentato in alcuni esempi. Il comando si puo' usare sia in modo immediato che in un programma. Esso si scrive:

```
POKE indirizzo byte,numero
```

sia l'indirizzo del byte che il numero possono essere delle variabili. Si ha la limitazione che l'indirizzo del byte deve essere esistente nel calcolatore ed il numero non deve superare 255, infatti un byte non puo' contenere un numero piu' grande.

Sappiamo anche leggere il contenuto di un byte usando il comando PEEK; esso si scrive:

```
PEEK(indizzo byte o variabile contenente l'indirizzo)
```

e la lettura fornisce un numero minore o uguale a 255.

Ricordiamo che le 506 posizioni dello schermo hanno nella memoria del VIC una immagine in 506 bytes consecutivi a partire dal byte di indirizzo 7680.

Dopo aver rimesso il VIC nelle condizioni iniziali, scrivete e provate il seguente programma:

```
10 REM SCRITTURA CON POKE
15 CD$=CHR$(17)
20 FOR K=1 TO 66
30 POKE 7679+K,65
40 NEXTK
45 PRINT CD$;CD$;CD$;CD$;
50 PRINT "HO SCRITTO 66 PICCHE"
60 STOP
```

vedrete apparire sulla quinta riga HO SCRITTO 66 PICCHE ma nelle prime 3 righe dove pensavate di avere scritto 66 picche queste non ci sono.

Abbiamo creato la variabile CD$ con il codice del CRSR in giu' per spostare il cursore in basso e non disturbare le righe dello schermo dove abbiamo scritto con le POKE quando scriviamo il messaggio con la PRINT. Infatti le POKE non spostano il cursore.

Aggiungete ora al programma le linee che seguono:

```
60 FOR K=1 TO 66
70 PRINT PEEK(7679+K);
80 NEXTK
90 STOP
```

e fate girare di nuovo il programma. Vedrete apparire sullo schermo le prime 4 linee bianche, poi il messaggio HO SCRITTO 66 PICCHE e poi 66 volte il numero 65 che e' il codice del carattere grafico "picche".

Come mai non si vedono le picche che pure ci sono? Ora lo capiremo.

IL video ha nella memoria del VIC due immagini, una, quella che inizia in 7680, serve per scrivere i codici dei caratteri, l'altra, che inizia in 38400 serve per il colore. Quando si fa la pulizia dello schermo nella mappa dei caratteri (da 7680 in poi) va il numero 32 che e' il codice dello spazio, mentre nella mappa dei colori (da 38400 in poi) va il codice del colore dello sfondo, pero' nella forma "modulo 16" (per avere il numero del colore dovete dividere per 16 e tenere il resto). I numeri colore sono i primi 8 visti per la combinazione sfondo schermo e cioe' il numero del tasto colore corrispondente -1.

Provate a sostituire nel programma precedente la linea 70 con la seguente:

```
70 POKE 38399+K,2
```

e fate girare il programma. Ora vedrete apparire nelle prime 3 righe dello schermo i 66 picche in rosso, infatti abbiamo usato per il colore il numero 2 che corrisponde al rosso. Per completare l'argomento aggiungete al vostro programma le linee seguenti e fatelo girare:

```
90 FOR K=1 TO 66
93 PRINT PEEK(38399+K);
95 NEXTK
98 STOP
```

vedrete apparire 66 numeri, osservateli bene e vedrete che essi sono tutti uguali a 2 modulo 16. Questi numeri sono i

contenuti della prime 3 righe della mappa colore dello schermo.

Potete ora riprovare il programma precedente aggiungendo nella linea 30 il numero 128 al codice del carattere. Avete percio':

```
30 POKE 7679+K,65+128
```

facendo girare il programma vedrete i picche in campo inverso. Infatti i codici dei caratteri in campo inverso si ottengono dagli altri aggiungendo 128 come e' spiegato anche nella Appendice B.

Nell'esempio di programma appena visto si sono usate le solite tecniche dei cicli e delle PRINT con i caratteri di controllo.

Riepilogando, vediamo le regole per scrivere direttamente sul video con le POKE:

. si deve scrivere una POKE per posizionare il carattere nella mappa caratteri del video che va dal byte 7680 al byte 8185;

. si deve scrivere una POKE per attivare il colore voluto nella posizione corrispondente nella mappa colore del video che va dal byte 38400 al byte 38905.

Se ripensate agli esempi colore del precedente paragrafo potete concludere che la PRINT produce il colore servendosi del colore del cursore, ma agisce sia sulla mappa caratteri che su quella colori del video.

Nella pagina seguente riportiamo le due mappe del video; esse possono essere utili per preparare programmi ottenendo grafica a colori con l'uso delle POKE.

M A P P A   C A R A T T E R I

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
7680	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7702	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7724	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7746	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7768	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7790	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7812	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7834	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7856	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7878	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7900	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7922	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7944	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7966	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7988	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8010	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8032	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8054	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8076	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8098	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8120	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8142	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
8164	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

M A P P A   C O L O R I

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
38400	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38422	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38444	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38466	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38488	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38510	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38532	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38554	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38576	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38598	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38620	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38642	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38664	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38686	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38708	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38730	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38752	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38774	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38796	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38818	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38840	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38862	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
38884	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.

3.4. ANCORA UN ESERCIZIO SUL COLORE

Provate ora un programma nel quale si usa dapprima il comando PRINT e poi il comando POKE per far apparire 462 palline colorate sullo schermo.

```
  1 PRINT CHR$(147);CHR$(31);
  5 P=1
 10 C$=CHR$(144)+CHR$(5)+CHR$(28)+CHR$(159)
 15 C$=C$+CHR$(156)+CHR$(30)+CHR$(31)+CHR$(158)
 17 POKE 36879,27
 20 PRINT"PROVA CON LE PRINT"
 25 FOR K=0 TO 461
 30 PRINT MID$(C$,P,1);CHR$(113);
 40 P=P+1
 42 IF P = 9 THEN P=1
 50 NEXTK
 55 FOR T=1 TO 1000:NEXTT
 60 PRINT CHR$(147);CHR$(31);
 63 POKE 36879,27
 65 PRINT"PROVA CON LE POKE"
 70 FOR G=0 TO 461
 75 POKE 7702+G,81
 80 NEXTG
 83 K=0
 85 FOR G=0 TO 461
 90 POKE 38422+G,K
 95 K=K+1
 97 IF K = 8 THEN K=0
 99 NEXTG
100 FOR T=1 TO 1000:NEXTT
110 GOTO1
```

La linea 1 pulisce lo schermo (CHR$(147)) e attiva il colore blu per il cursore (CHR$(31)).
La linea 5 pone la variabile P=1, tale variabile serve per far variare il numero che fa da puntatore al colore desiderato nella stringa C$.
Le linee 10 e 15 fabbricano la stringa C$ con i caratteri colore.
La linea 17 pone il colore sfondo schermo alla situazione dell'accensione.
La linea 20 scrive sulla prima riga dello schermo quale lavoro viene svolto.
La linea 25 inizia un ciclo di programma che viene eseguito fino a quando K, partendo dal valore 0, raggiunge il valore 461; le istruzioni eseguite vanno dalla linea 30 alla linea 50.
La linea 30 stampa il carattere colore che preleva (in base a P) dalla stringa C$ e la pallina colorata senza andare a capo.

La linea 40 incrementa di 1 la variabile P.
La linea 45 controlla il valore raggiunto da P e se esso e' diventato 9 lo rimette al valore 1.
La linea 55 crea un ritardo basato sul numero 1000 per consentire la permanenza delle immagini.
La linea 60 agisce come la linea 1.
La linea 63 agisce come la linea 17.
La linea 65 scrive sulla prima riga dello schermo quale lavoro viene svolto.
La linea 70 inizia un ciclo, che termina alla linea 80, controllato dalla variabile G e nel quale lo schermo viene riempito dalla seconda riga alla penultima di palline, che pero' non appaiono, dato che i byte colore non sono ancora scritti.
La linea 83 pone la variabile K=0: tale variabile serve a definire il numero colore da usare con il comando POKE per far apparire le palline, gia' scritte, in colore.
La linea 85 inizia un ciclo, controllato dalla variabile G per scrivere con il comando POKE nei byte della mappa colore dello schermo dall'inizio della seconda riga fino a tutta la penultima facendo alternare i colori. Il ciclo termina alla linea 99.
Le linee 95 e 97 servono per incrementare la variabile K e controllare che il suo valore non superi 7.
La linea 100 crea il solito ciclo di attesa.
La linea 110 fa ricominciare il programma.

Quando avete valutato quale dei due metodi di porre palline colorate sullo schermo e' il piu' veloce, potete premere il tasto STOP RUN ed interrompere il programma.

CAPITOLO 4

IL MOVIMENTO

4.1. FACCIAMO VOLARE UN UCCELLINO

Vi proponiamo un programma con il quale vedrete volare un uccellino, cioe' otterrete il movimento detto anche "animazione" sul video. Fino ad ora siamo ricorsi alla funzione CHR$, inserita nei precedenti programmi, per definire ed ottenere i caratteri di controllo che non possiamo stampare facilmente nel libro. D'ora in poi potremo anche segnalare i tasti da premere racchiudendoli tra i simboli di minore e maggiore. Per esempio, per indicare di premere SHIFT e R, scriveremo <SHIFT e R>.

Ora seguite come al solito le istruzioni:

. scrivete NEW e premete RETURN;

. premete contemporaneamente STOP RUN e RESTORE;

. scrivete il programma che segue premendo RETURN dopo ogni linea:

```
10 PRINT "<SHIFT e CLR HOME><SHIFT e J><SHIFT e Q>
         <SHIFT e K>"
20 FOR T=1 TO 150:NEXTT
30 PRINT "<SHIFT e CLR HOME><SHIFT e U><SHIFT e Q>
         <SHIFT e I>"
40 FOR T=1 TO 150:NEXTT
50 GOTO 10
```

(la linea 10 e la linea 30 sembrano lunghe, ma premendo i tasti vengono visualizzati solo 4 caratteri tra gli apici). Quindi scrivete RUN e premete RETURN.

Vedrete nell'angolo in alto a sinistra un uccellino che muove le ali. In realta' il movimento e' un illusione ottica.

Vediamo cosa fa il programma:

. la linea 10 esegue una PRINT di una stringa che contiene: il carattere di pulizia video (corrisponde ad un cuore in campo inverso), il carattere che disegna l'ala sinistra dell'uccellino curva verso l'alto, il carattere che disegna

il corpo dell'uccellino, il carattere che disegna l'ala destra dell'uccellino curva verso l'alto;

. la linea 20 e la linea 40 creano una attesa, basata sul numero 150, per far permanere l'immagine sullo schermo il tempo necessario a vederla;

. la linea 30 esegue una PRINT analoga a quella della linea 10, solo che l'uccellino ha le ali voltate verso il basso;

. la linea 50 rimanda alla 10 e quindi i due disegni dell'uccellino con le ali curve verso l'alto e verso il basso si alternano continuamente sul video dando l'illusione del movimento.

Se volete fermare il programma premete STOP RUN.

Potete provare a modificare nelle linee 20 e 40 il numero 150. Diminuendolo le ali si muoveranno piu' velocemente, mentre aumentandolo il movimento viene rallentato.

Nell'esempio precedente il nostro uccellino si limita a muovere le ali restando sempre nello stesso posto. Se provate il programma che segue lo vedrete anche muoversi nello schermo. Per ottenere lo spostamento si usano i caratteri di spostamento del cursore che creiamo come stringhe all'inizio del programma chiamando CR$ la stringa che fa spostare a destra, CL$ la stringa che fa spostare a sinistra, CD$ la stringa che fa spostare in giu'.

Provate dunque questo programma:

```
10 CR$="<CRSR laterale>"
20 CL$="<SHIFT e CRSR laterale>"
30 CD$="<CRSR verticale>"
40 PRINT"<SHIFT e CLR HOME>";
50 PRINT CD$;CR$;"<SHIFT e J><SHIFT e Q>
         <SHIFT e K>";CL$;CL$;CL$;
60 FOR T=1 TO 150:NEXTT
70 PRINT " <SHIFT e U><SHIFT e Q><SHIFT e I> ";
         CL$;CL$;CL$;
80 FOR T=1 TO 150:NEXTT
90 PRINT CL$;"    ";CL$;CL$;CL$;
95 FOR T=1 TO 150: NEXTT
97 GOTO 50
```

e vedrete l'uccellino che vola spostandosi nello schermo. Per fermare il programma premete STOP RUN.

Vediamo come si ottiene il movimento:

. le linee da 10 a 30 creano le stringhe con i caratteri di

controllo per il cursore da usare nelle PRINT;

. la linea 40 pulisce lo schermo;

. la linea 50 produce: uno spostamento in giu', uno spostamento a destra, il disegno dell'uccellino con le ali in su', 3 spostamenti verso sinistra, riposizionandosi sull'uccellino;

. le linee 60, 80 e 95 creano le attese necessarie per percepire i disegni;

. la linea 70 produce: uno spazio, il disegno dell'uccellino con le ali in giu', uno spazio , 3 spostamenti verso sinistra, cioe' si riposiziona quasi sull'uccellino, manca una posizione;

. la linea 90 produce: l'ultimo spostamento a sinistra per sovrapporsi all'uccellino, 3 spazi per cancellare l'uccellino, 1 spazio, 3 spostamenti a sinistra;

. la linea 97 rimanda alla prima PRINT della figura dell'uccellino che viene ora ridisegnato spostato a destra.

Abbiamo imparato ad ottenere il movimento servendoci del comando PRINT del BASIC. Il movimento si puo' anche ottenere usando i comandi PEEK e POKE e lo vedremo nel prossimo paragrafo.

4.2. IL MOVIMENTO USANDO I COMANDI DIRETTI

Il video ha due immagini nella memoria del VIC, la prima e' la mappa dei caratteri ed occupa 506 posizioni dal byte di indirizzo 7680 al byte di indirizzo 8185, la seconda e' la mappa dei colori ed occupa ancora 506 posizioni dal byte di indirizzo 38400 al byte di indirizzo 38905. Quando si premono contemporaneamente i tasti STOP RUN e RESTORE o SHIFT e CLR HOME, lo schermo si ripulisce. Nel primo caso si torna anche ai colori iniziali di bordo azzurro, schermo bianco e cursore blu, nel secondo i colori possono anche essere diversi e dipendono dalla situazione al momento del comando.

Se vogliamo localizzare una posizione dello schermo e indichiamo con Y una delle possibili 23 righe partendo da 0 per la prima in alto, e con X una delle possibili 22 colonne partendo da 0 per la prima di sinistra, dobbiamo usare la formura seguente:

P = 7680 + X + 22 * Y

per ottenere la posizione del carattere nella mappa dei caratteri, e la formula seguente:

C = 38400 + X + 22 * Y

per ottenere la posizione del colore nella mappa dei colori.

Nella Appendice C trovate riportate le due mappe.

Il valore P serve per scrivere con un comando POKE il codice del carattere voluto nella posizione voluta, e il valore C serve per scrivere il codice del colore voluto nella posizione voluta per quel carattere. Ricordiamo che i codici dei colori vanno da 0 a 7 cioe' se si fa riferimento al tasto colore si deve togliere 1 al numero scritto sopra il tasto.

Proviamo ora a mettere una pallina bianca al centro del video; dobbiamo procedere cosi':

. X=10 e Y=10 per il punto centrale;

. P=7680+10+22*10=7910   C=38400+10+22*10=38630;

. scriviamo ed eseguiamo: POKE 36879,8 per ottenere tutto lo schermo nero;

. scriviamo ed eseguiamo: POKE 7910,61 e con questo poniamo una pallina bianca nel centro del video;

. scriviamo ed eseguiamo: POKE 38630,2 e con questo cambiamo il colore della pallina in rosso.

Dopo aver riepilogato questi concetti possiamo scrivere e provare il programma che segue, dopo aver rimesso il calcolatore nelle condizioni iniziali premendo contemporaneamente STOP RUN e RESTORE, ed aver ripulito la memoria scrivendo NEW e premendo RETURN:

```
 10 PRINT"<SHIFT e CLR HOME>"
 20 POKE 36879,9
 30 POKE 36878,15
 40 X=1
 50 Y=1
 60 DX=1
 70 DY=1
 80 POKE 7680+X+22*Y,81
 90 FOR T=1 TO 10:NEXTT
100 POKE 7680+X+22*Y,32
110 X=X+DX
120 IF X=0 OR X=21 THEN DX=-DX:POKE 36876,220
130 Y=Y+DY
```

```
140 IF Y=0 OR Y=22 THEN DY=-DY:POKE 36876,230
150 POKE 36876,0
160 GOTO80
```

Vedrete una pallina bianca che si muove su uno schermo tutto nero con il bordo in bianco. Quando la pallina arriva sulla cornice rimbalza e cambia direzione. Vediamo cosa fanno le diverse linee di programma:

. La linea 10 pulisce lo schermo, la linea 20 produce lo sfondo nero con il bordo bianco, la linea 30 predispone il volume per il suono.

. Le linee da 40 a 70 predispongono il valore iniziale 1 per X e Y, posizione iniziale della pallina, ed il valore iniziale 1 per i due incrementi DX e DY. Se l'incremento e' positivo per Y, la pallina si sposta in basso, mentre se e' negativo si sposta in alto. Analogamente per X, l'incremento positivo provoca uno spostamento a destra, quello negativo a sinistra.

. La linea 80 fa apparire la pallina bianca nella posizione corrispondente ai valori di X e Y. Infatti la mappa colore dello schermo e' bianca per effetto dello SHIFT e CLR HOME alla linea 10 e la stampa avviene in modo diretto.

. La linea 90 crea un breve ritardo per consentire di vedere la pallina.

. La linea 100 cancella la pallina dalla precedente posizione.

. La linea 110 modifica il valore di X e la linea 120 se X ha raggiunto uno dei due bordi laterali predispone l'incremento DX con segno contrario per invertire la direzione del movimento laterale. Inoltre la linea 120 produce un suono se la pallina ha toccato.

. La linea 130 modifica il valore di Y e la linea 140 se Y ha raggiunto uno dei due bordi orizzontali predispone il cambio di segno dell'incremento DY per invertire la direzione del movimento verticale. Inoltre la linea 140 produce un suono se la pallina ha toccato.

. La linea 150 smorza il suono precedente e la linea 160 fa ritornare al movimento della pallina, cioe' a disegnare la pallina nel nuovo punto.

Quando siete stanchi di vedere rimbalzare la vostra pallina premete RUN STOP e RESTORE insieme, poi scrivete le istruzioni che seguono, esse vanno ad aggiungersi al programma gia' in memoria:

```
32 FOR L=1 TO 10
35 POKE 7680+INT(RND(1)*506),102
37 NEXTl
155 IF PEEK(7680+X+22*Y)=102 THEN DX=-DX:DY=-DY:
    POKE 36876, 180:GOTO 110
```

Se ora fate girare il programma vedrete degli ostacoli in posizioni casuali dello schermo e vedrete che la pallina rimbalza quando li incontra. Le linee da 32 a 37 producono gli ostacoli, e la linea 155 fa invertire la marcia alla pallina quando incontra gli ostacoli.

Con l'uso delle funzioni PEEK e POKE si possono scrivere dei programmi molto interessanti, tali programmi si otterrebbero con maggior fatica usando il comando PRINT.

Se volete veder cambiare i colori mentre la pallina si muove, scrivete le due istruzioni che seguono:

```
155 P=INT(RND(1)*255)
140 IF Y=0  OR  Y=22  THEN  DY=-DY:POKE  36876,230:POKE
36879,P
```

e poi cancellate le istruzioni 32, 35 e 37 scrivendo ogni numero di linea seguito da RETURN. Provate ora di nuovo a fare girare il programma.

CAPITOLO 5

IL SUONO

5.1. FACCIAMO MUSICA

Nel capitolo precedente abbiamo gia' sperimentato degli effetti sonori nel gioco della pallina che rimbalza. Provate ora il programma che segue:

```
 10 PRINT "<SHIFT e CLR HOME>"
 20 FOR I=1 TO 5
 30 POKE 36873+I,0
 40 NEXTI
 50 PRINT "QUALE TIMBRO (1-4)?"
 60 PRINT "PER FINIRE 0"
 70 INPUT N
 80 IF N=0 THEN END
 90 PRINT "QUALE TONALITA' (128-254)?"
100 INPUT P
110 POKE 36878,14
120 POKE 36873+N,P
130 FOR J=1 TO 2000:NEXTJ
140 GOTO 10
```

facendolo girare sentirete dei suoni.

Anche questa volta alcuni comandi POKE hanno prodotto effetti speciali.

Vediamo quali sono le possibilita' sonore del VIC. Il calcolatore dispone di 5 locazioni di memoria per controllare i suoni. Queste locazioni sono i byte di indirizzo:

36874, 36875, 36876, 36877, 36878.

Il loro utilizzo e' il seguente:

POKE 36878,X — predispone il volume del suono, X puo' variare da 0 a 15; dato che il semibyte di ordine piu' elevato puo' essere impegnato per il multicolore, fate attenzione;

POKE 36874,X — predispone la prima tonalita' e

X puo' variare da 128 a 255;

POKE 36875,X predispone la seconda tonalita' e X puo' variare da 128 a 255;

POKE 36876,X predispone la terza tonalita' e X puo' variare da 128 a 255;

POKE 36877,X predispone il generatore di rumore bianco e X puo' variare da 128 a 255.

I suoni non sono udibili fino a quando il volume non raggiunge un valore sufficiente; si deve provare perche' si hanno differenze passando da un amplificatore all'altro.

5.2. LE NOTE MUSICALI

Riportiamo il valore da dare alla tonalita' per ottenere le note musicali:

NOTA	VALORE	NOTA	VALORE
DO	135	SOL	215
DO#	143	SOL#	217
RE	147	LA	219
RE#	151	LA#	221
MI	159	SI	223
FA	163	DO	225
FA#	167	DO#	227
SOL	175	RE	228
SOL#	179	RE#	229
LA	183	MI	231
LA#	187	FA	232
SI	191	FA#	233
DO	195	SOL	235
DO#	199	SOL#	236
RE	201	LA	237
RE#	203	LA#	238
MI	207	SI	239
FA	209	DO	240
FA#	212	DO#	241

Come abbiamo visto nel precedente paragrafo 3 diversi byte controllano la tonalita'; la piu' bassa dipende dal byte 36874, la media dal byte 36875 e la alta dal byte 36876. Ponendo in ciascuno di questi byte il valore corrispondente alle note si ottiene il suono corrispondente, purche' il volume (byte 36878) sia alla soglia giusta. Ponendo invece un valore nel byte 36877 si ottiene un rumore bianco.

Provate a porre il volume in modo che possiate sentire qualcosa con il vostro televisore e poi provate a generare delle note usando in modo diretto i comandi POKE.

Il VIC puo' produrre per ogni tonalita' 128 note diverse.

5.3. LA MUSICA PROGRAMMATA

Vediamo come si puo' programmare un pezzo di musica. Dovete scrivere sotto forma di tabellina, di tante righe quanto e' necessario, e di 2 colonne, il vostro pezzo di musica, mettendo nella prima colonna il codice della nota e nella seconda la durata della medesima, per esempio, cosi':

CODICE NOTA	DURATA
225	360
225	360
225	240
228	120
231	360
231	240
228	120
231	360
231	240
228	120
231	240
232	120
235	720
240	360
235	360
231	360
225	360
235	240
232	120
231	240
228	120
225	480

Se per la durata avete dei dubbi, potrete fare vari tentativi e corregerla dopo aver fatto delle prove.

Questa tabella di valori viene caricata in memoria sfruttando il comando del BASIC DATA. Cioe' si prepara un blocco di dati usando le righe della tabellina una dopo l'altra.

Alla fine per chiudere i dati della melodia si usa il numero -1. Il programma preléva dal blocco dati ordinatamente ogni nota e la sua durata e se ne serve per produrre la musica.

Il blocco dati da scrivere per la melodia di cui sopra e' il seguente:

```
1000 DATA 225, 360, 225, 360, 225, 240
1010 DATA 228, 120, 231, 360, 231, 240
1020 DATA 228, 120, 231, 240, 232, 120
1030 DATA 235, 720, 240, 360, 235, 360
1040 DATA 231, 360, 225, 360, 235, 240
1050 DATA 232, 120, 231, 240, 228, 120
1060 DATA 225, 480
1070 DATA -1
```

Le istruzioni DATA partono da 1000, mentre il programma

che segue ha numeri di linea piu' bassi; il programma va bene per suonare qualunque melodia modificando solo il blocco dei DATA. Segue il programma:

```
  1 PRINT "<SHIFT e CLR HOME>FACCIAMO MUSICA"
  5 PRINT "BUON ASCOLTO"
 10 S2=36875
 20 V=36878
 30 REM CICLO PER SUONARE
 40 POKE V,15
 50 READ P
 60 IF P= -1 THEN 200
 70 READ D
 80 POKE S2,P
 90 FOR N=1 TO D:NEXT N
100 POKE S2,0
110 FOR N=1 TO 20: NEXT N
120 GOTO 50
200 REM FINISCE LA MUSICA
210 END
```

Alle linee 1 e 5 si manda un messaggio al video. La linea 10 pone in S2 l'indirizzo del byte da usare per la tonalita' e la linea 20 pone in V l'indirizzo del byte volume. La linea 40 mette 15 nel volume. La linea 50 legge il valore del blocco dati al quale e' posizionato il puntatore interno (tale puntatore e' prima del primo dato al momento del RUN e si sposta ogni volta che si legge un dato dal blocco) e lo pone in P. La linea 60 se trova P=-1 va alla 200, dove termina il programma. La linea 70 legge in D la durata della nota appena letta in P. La linea 80 suona la nota P e con la linea 90 la fa durare secondo D. La linea 100 spegne la nota e la mantiene nulla per una durata controllata dal numero 20 alla linea 110. La linea 120 rimanda al ciclo del suono.

Provate ora il programma, poi provate a sostituire al blocco dati in 1000 un altro blocco e riprovate. Vi proponiamo le seguenti melodie:

```
1000 DATA 217, 400, 213, 400, 223, 400
1010 DATA 227, 200, 234, 200, 230, 400
1020 DATA 227, 200, 234, 200, 230, 400
1030 DATA 223, 400, 227, 400, 217, 400
1040 DATA 213, 600
1050 DATA -1
```

```
1000 DATA 225, 1000, 228, 1000, 231, 1000
1010 DATA 232, 1000, 235, 1000, 237, 1000
1020 DATA 239, 1000, 240, 1000
1030 DATA -1
```

5.4. IL PIANOFORTE

Ora se volete potete provare ad usare il VIC come un pianoforte. Si usa questo accorgimento:

. si fanno corrispondere le note musicali che sono necessarie per produrre il pezzo di musica desiderato ad alcuni tasti; se bastano 8 note diverse si puo' procedere come nell'esempio riportato, dove bastano 8 note che si fanno corrispondere ai tasti numerici da 1 a 8. Se si desidera poter sfruttare le ottave in tutta la loro estensione si possono usare anche altri e piu' numerosi tasti che abbiano codifica ASCII crescente in sequenza, per esempio A,B,C,ecc, e programmare un semplice calcolo per risalire al numero che serve da indice per reperire il valore numerico corrispondente a ciascuna nota.

Nel nostro esempio si hanno:

1	2	3	4	5	6	7	8
SI	DO#	RE#	MI	FA#	SOL#	LA#	SI

Provate il programma che segue:

```
 10 REM MEMORIZZA INDIRIZZI REGISTRI
 20 S2=36875
 30 V=36878
 40 POKE S2,0
100 REM MEMORIZZA LA SCALA MUSICALE
110 FOR N=1 TO 8
120 READ A(N)
130 NEXT N
140 DATA 223, 227, 230
150 DATA 231, 234, 236
160 DATA 238, 239
200 REM CICLO PER SUONARE
210 POKE V,10
220 GET A$: IF A$="" THEN 220
230 N=VAL(A$)
240 IF N=0 OR N=9 THEN 300
250 POKE S2,0
260 FOR T=1 TO 25:NEXT T
270 POKE S2,A(N)
280 GOTO 220
300 REM FINE PROGRAMMA
310 POKE S2,0
320 END
```

Le linee da 10 a 40 memorizzano nelle variabili S2 e V gli

indirizzi della tonalita' e del volume, e viene posta a zero la tonalita'.

Le linee da 100 a 130 memorizzano i valori corrispondenti alla scala musicale nella variabile con indice A, che non e' stata dimensionata avendo solo 8 elementi. I valori delle note vengono prelevati dal blocco dati generato dai DATA alle linee 140-160.

La linea 210 pone il volume a 3. La linea 220 chiede una nota e se non si risponde resta in attesa. Qui dovete premere il tasto numerico corrispondente alla nota desiderata. La linea 230 calcola il valore del tasto ed ottiene un numero da 1 a 8 che serve come indice per andare a prelevare da A(N) il codice della nota. Se premete 0 o 9 il programma termina, questa analisi viene fatta alla linea 240.

La linea 270 suona la nota voluta, mentre la 250 e la 260 creano un intervallo di silenzio. La durata del suono dipende da voi, dalla velocita' con la quale premete i tasti.

Provate il programma, se volete provate a suonare queste musichette:

```
1  1  5  5  6  6  5
4  4  3  3  2  2  1
5  5  4  4  3  3  2
5  5  4  4  3  3  2
1  1  5  5  6  6  5
4  4  3  3  2  2  1  8
9


3  3  4  5  5  4  3  2
1  1  2  3  3  2  2
3  3  4  5  5  4  3  2
1  1  2  3  2  1  1
0
```

Buon divertimento!

CAPITOLO 6

CONOSCIAMO MEGLIO IL VIC

6.1. STRUTTURA DI UN CALCOLATORE ELETTRONICO

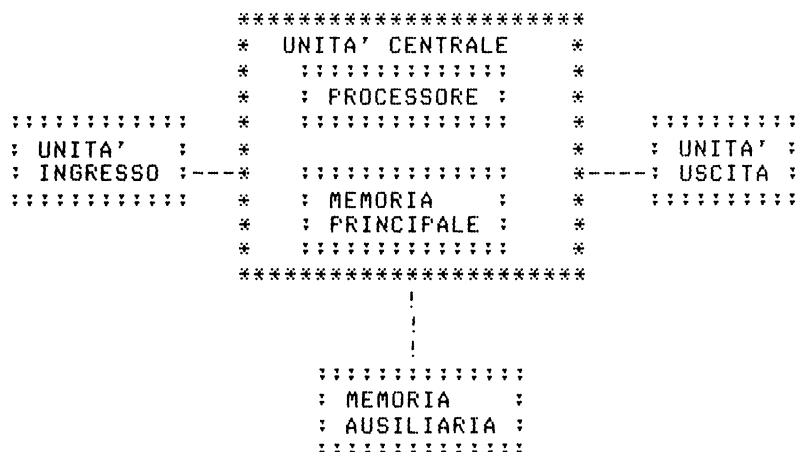
Il nome completo e': calcolatore elettronico automatico digitale. Analizziamo il significato delle singole parole:

. calcolatore: esegue calcoli;
. elettronico: costruito usando tecniche elettroniche;
. automatico: funziona automaticamente;
. digitale: elabora informazioni codificate in cifre (digit significa cifra in inglese).

Le principali componenti un calcolatore elettronico sono:

. Unita' Centrale (CPU da Central Processing Unit);
. Unita' di ingresso (INPUT);
. Unita' di uscita (OUTPUT);
. Memoria secondaria di massa.

Riportiamo uno schema semplificato:

```
                  **********************
                  *  UNITA' CENTRALE   *
                  *   ::::::::::::::   *
                  *   : PROCESSORE :   *
::::::::::::      *   ::::::::::::::   *     ::::::::::
: UNITA'   :      *                    *     : UNITA' :
: INGRESSO :---*   ::::::::::::::   *----: USCITA :
::::::::::::      *   : MEMORIA    :   *     ::::::::::
                  *   : PRINCIPALE :   *
                  *   ::::::::::::::   *
                  **********************
                            !
                            !
                            !
                       ::::::::::::::
                       : MEMORIA    :
                       : AUSILIARIA :
                       ::::::::::::::
```

Il PROCESSORE comprende: . Unita' di Controllo;
. Unita' Aritmetico Logica;
. Registri Speciali.

La MEMORIA PRINCIPALE comprende: . Memoria ROM (Read Only Memory) a sola lettura;
. Memoria RAM (Random Ac= cess Memory) per lettu= ra e scrittura.

L'UNITA' DI INGRESSO e' generalmente una tastiera.

L'UNITA' DI USCITA e' generalmente un video o una stampan= te.

La MEMORIA AUSILIARIA e' un mezzo magnetico tipo nastri o dischi.

L'UNITA' DI CONTROLLO governa il sistema servendosi degli automatismi interni e dei programmi che sono permanentemente memorizzati in ROM o temporaneamente memorizzati in RAM. Al funzionamento concorre per la sua parte l'UNITA' ARITMETICO LOGICA ed hanno una funzione determinante i REGISTRI SPECIALI.

La memoria ROM contiene i programmi forniti dalla casa costruttrice necessari per fare funzionare il sistema. Nei calcolatori Personal solitamente in ROM risiedono il Sistema Operativo e l'Interprete Basic.

La memoria RAM e' a disposizione dell'utente per memorizzare i suoi programmi. Una parte di RAM e' usata come area di lavoro del sistema.

Ogni calcolatore e' costruito in modo tale che puo' riconoscere ed eseguire un gruppo di istruzioni elementari. Queste istruzioni costituiscono il LINGUAGGIO MACCHINA.

Servendosi delle istruzioni elementari si scrivono i programmi, che sono sequenze di istruzioni. I programmi devono essere memorizzati nella memoria del calcolatore, servendosi di una unita' di ingresso. Esiste un automatismo di funzionamento che consente al calcolatore di prelevare una dopo l'altra le istruzioni dalla memoria e di eseguirle. A questo automatismo provvede l'unita' di controllo.

Nella memoria del calcolatore vengono conservati anche i dati sui quali si deve lavorare e i risultati delle elaborazioni. I dati ed i risultati vengono comunicati servendosi di una unita' di uscita.

6.2. STRUTTURA DELLA MEMORIA

La memoria del calcolatore puo' essere immaginata come formata da una serie di caselline dotate di un indirizzo di riconoscimento. Ogni casellina (BYTE) puo' contenere una determinata quantita' di informazioni. Nella pratica si usa chiamare byte sia il contenitore che il contenuto e si parla di indirizzo del byte.

Esiste una unita' di misura per le memorie denominata K. Si fa corrispondere K a 1024; dicendo che un calcolatore ha 4K di memoria, si intende che ha 4*1024 byte di memoria.

Gli indirizzi della memoria partono da 0, per cui 1K di memoria corrisponde a indirizzi da 0 a 1023.

Questa organizzazione della memoria e' valida sia per la memoria ROM che per la memoria RAM, la vera differenza tra i due tipi di memoria sta nel fatto che quando si spegne il calcolatore la RAM si cancella e la ROM no. Inoltre l'utente non puo' scrivere nella ROM, ma solo nella RAM.

Ogni byte e' in realta' formato da 8 unita' chiamate singolarmente bit. Ogni bit puo' contenere una informazione elementare che si fa convenzionalmente corrispondere a 0 o 1. La logica del calcolatore e' quindi di tipo binario, a livello elementare sono possibili solo due simboli diversi, lo zero e l'uno. I possibili contenuti di un byte si ottengono facendo ruotare in tutti i modi possibili 0 e 1 nelle sue posizioni, riuscendo ad ottenere 256 combinazioni diverse passando da 8 zeri ad 8 uno consecutivi.

Volendo comprendere meglio come funziona il calcolatore bisogna fare un cenno all'aritmetica binaria.

6.3. ARITMETICA BINARIA

Consideriamo un qualunque numero come:

```
4875      in esso: 4 rappresenta le migliaia;
                   8 rappresenta le centinaia;
                   7 rappresenta le decine;
                   5 rappresenta le unita'.
```

Possiamo quindi scrivere:

4875 = 4x1000+8x100+7x10+5

cioe' ogni cifra assume il valore che dipende dalla sua posizione nel numero. Questo valore viene calcolato servendosi delle potenze del 10; siamo infatti nel sistema

numerico decimale. La cifra delle unita' ha il valore che si ottiene moltiplicandolo per 10 elevato a 0, cioe' per 1 (qualunque numero elevato a zero da' uno).
Ragionamenti analoghi valgono per le cifre dopo la virgola, in questo caso le potenze del 10 hanno esponente negativo.

Per costruire un sistema di numerazione non decimale si possono usare le stesse convenzioni usate per il sistema decimale. Quello che cambia e' il numero di simboli diversi usati per le cifre e la base che con le sue potenze attribuisce valore alle cifre.

Per costruire il sistema di numerazione binario si usano come cifre i due simboli 0 e 1 e come base il numero 2. Un numero binario si presenta pertanto cosi':

1010110.

Per calcolarne il valore dobbiamo procedere cosi':

. il numero e' formato da 7 cifre, quindi dobbiamo considerare le prime 7 potenze di 2, da 2 elevato a 0 a 2 elevato a 6; esse sono in ordine descrescente:

64 32 16 8 4 2 1

. per ottenere il valore decimale del numero binario 1010110 moltiplichiamo le cifre per le corrispondenti potenze del 2 cosi':

1x64+0x32+1x16+0x8+1x4+1x2+0x1 ed otteniamo:

64 + 16 + 4 + 2 = 86

quindi il numero binario 1010110 corrisponde al numero decimale 86. Sono state necessarie piu' cifre binarie perche' la base 2 e' piu' piccola della base 10.

Nella memoria del nostro calcolatore un byte contiene 8 bit; vediamo quanto vale un numero formato da otto bit 1. Dobbiamo aggiungere una potenza a quelle viste prima:

128 64 32 16 8 4 2 1.

Quindi ricalcoliamo: 128+64+32+16+8+4+2+1=255.

Possiamo concludere che 1 byte puo' contenere al massimo il numero decimale 255.

Si usa numerare le posizioni dei bit nel byte in base alla potenza del 2 che corrisponde alla loro posizione:

```
bit    7    6    5    4    3    2    1    0
     -----------------------------------------
     !    !    !    !    !    !    !    !    !
     !    !    !    !    !    !    !    !    !
     -----------------------------------------
              Byte
```

Vediamo quale numero puo' essere contenuto in due byte associati, letti uno dopo l'altro. Per considerare il loro valore globale:

. dobbiamo aggiungere alla sinistra delle potenze gia' usate per valorizzare 1 byte le altre 8:

32768 16384 8192 4096 2048 1024 512 256 128 64 32 16 8 4 2 1

. se ora consideriamo sedici cifre 1 in sequenza il loro valore corrisponde alla somma delle potenze di 2 sopra elencate, che vanno da 2 elevato a 0 a 2 elevato a 15, ed il risultato e': 65535.

In certi casi il calcolatore usa due byte consecutivi.

I numeri binari sono poco pratici e troppo lunghi da scrivere. Per questa ragione si usa leggerli e scriverli usando l'aritmetica esadecimale, in base 16. In questa aritmetica occorrono 16 simboli diversi per rappresentare i numeri e la base e' 16. Le 16 cifre esadecimali, che esponiamo scrivendo nella riga sottostante il corrispondente valore decimale, sono:

Cifre esade=cimali	F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0
Valori	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Per ottenere un numero esadecimale da uno binario basta raggruppare i bit 4 a 4 e sostituire ogni gruppo con la corrispondente cifra:

```
        1 1 1 1 1 1 1 1   corrisponde a
           F       F      in esadecimale.
```

Un byte e' formato da 2 cifre esadecimali.

Esiste una convenzione per poter rappresentare i numeri con il segno. Essa e' quella di usare il primo bit a

sinistra per il segno. Se questo bit e' 0 il numero e' positivo, se esso e' 1 il numero e' negativo. Naturalmente in tale modo per il numero si puo' usare 1 bit in meno e quindi si ottiene un valore minore. Se si considera il numero ottenuto, associando due byte e usando la convenzione del segno, si vede che il massimo valore raggiungibile e': 32767.

Inoltre i numeri negativi sono scritti nella forma del complemento a 2. Questo significa che per ottenere la configurazione di bit presenti in memoria per un numero negativo si deve calcolare, nel caso di 2 byte associati, 2 elevato a 15 e sottrargli il numero in valore assoluto. Si ottiene cosi' la configurazione dei bit (15 bit) ed a questi si deve aggiungere il bit 1 del segno a sinistra.

Vediamo per esempio come appare in memoria il numero intero -3:

```
      +3 corrisponde a 0000 0000 0000 0011

      2 elevato a 15 corrisponde a 1000 0000 0000 0000

        1000 0000 0000 0000-
        0000 0000 0000 0011=
      ------------------------
        0111 1111 1111 1101    configurazione dei bit
                               senza il segno

        1111 1111 1111 1101    configurazione dei bit
                               con il segno

         F    F    F    D      se letto in esadecimale.
```

6.4. IL VIC A GRANDI BLOCCHI

Il microprocessore MCS 6502 controlla il funzionamento del VIC; esso gestisce le operazioni del video, della tastiera, della cassetta e di eventuali periferiche addizionali.

Il suono e i colori sono gestiti dal microprocessore MCS 6560 (Video Interface Chip).

La speciale interfaccia RS232C consente il collegamento per trasmissione a distanza e tra calcolatori. Sono possibili collegamenti con altre interfacce reperibili sul mercato.

Nella Figura 6-1 e' riportato uno schema a blocchi del VIC e nella Figura 6-2 si vede il VIC aperto.

Le caratteristiche tecniche del VIC sono:

. memoria RAM 5K
. possibile espansione 32K

. linguaggio PET BASIC
. tasti 66
. tipo tastiera standard
. tasti funzione 4 (che diventano 12)
. simboli grafici 62
. risoluzione grafica 176 x 184
. suono
. colore (2 modi)
. possibilita' di collegare floppy singolo
. capacita' disco 170K
. interfaccia BUS seriale
. interfaccia RS232C

Il microprocessore 6502 controlla lo svolgimento delle operazioni servendosi del registro contatore del programma e dei circuiti appositi. Il registro contatore del programma contiene sempre l'indirizzo della istruzione (in linguaggio macchina) che deve essere eseguita. Tale istruzione viene prelevata dalla ROM e trasferita nell'unita' di controllo nel registro dell'istruzione. Il registro contatore del programma viene automaticamente incrementato per puntare alla prossima istruzione, l'incremento viene ricavato dall'istruzione in esecuzione. Inizia il ciclo di esecuzione che , a seconda del tipo di istruzione, puo' comportare prelevamenti di dati dalla memoria o memorizzazioni. Il microprocessore lavora sempre per mezzo degli indirizzi e servendosi dei registri adibiti ai trasferimenti. Esiste un canale (BUS) di comunicazione a 16 bit per il trasferimento degli indirizzi. Per il trasferimento dei dati vengono usate linee bidirezionali che operano su 8 bit. Il senso del trasferimento, lettura o scrittura in memoria, dipende dalla linea di abilitazione (Read/Write).

Esistono poi altre linee di controllo. Una linea serve per la temporizzazione delle operazioni del calcolatore (clock). Un'altra serve per inizializzare il calcolatore al momento dell'accensione predisponendo i contenuti dei registri che servono al sistema per la gestione dell'insieme. Una terza linea serve per il riconoscimento delle interruzioni ed il loro servizio.

Il VIA-6522 (Versatile Interface Adaptors) fornisce l'interfaccia per la tastiera, le porte utente, il BUS seriale e le porte per i giochi.

```
                    ::::::::::::::      ::::::::::::::
                    : Espansione :      : Schede     :
                    : memoria    :      : programmate:
                    : 3K/8K/16K  :      : inseribili :
                    ::::::::::::::      ::::::::::::::
                              !              !
 ::::::::::::::            :::::::::::::::::::::
 : Schede     :            :    Piastra        :
 : programmate:----!       :    controllo      :
 : inseribili :    !       :::::::::::::::::::::
 ::::::::::::::    !                       !
                   !                       !
 ::::::::::::::    !                       !
 : Espansione :    !                       !
 : memoria 3K :----!-----------------------!
 ::::::::::::::      :::::::::::           !
                     :Stampante:           !
                     :::::::::::           !
                        !                  !
:::::::::::  ::::::: ::::::::::            !
:Unita'   :  :Modem: : Disco  :            !
:Cassetta :  ::::::: : singolo:            !
:::::::::::    !     ::::::::::            !
    !          !        !                  !
    !          !        !                  !
*************************************************
*Interfaccia Porte   BUS seriale      Porte     *-----!
*registratore utente                  espansione*     !
*                                               *---! !
*                                               *   ! !
*                                        Porte  *-! ! !
*                  V I C                 giochi * ! ! !
************************************************* ! ! !
                               :::::::::::::::::  ! ! !
                               : Comandi giochi:--! ! !
                               : Penna ottica  :    ! !
                               :::::::::::::::::    ! !
                                                    ! !
                             :::::::::::::::::::    ! !
                             :  Alimentatore   :    ! !
                             :                 :----! !
                             :::::::::::::::::::      !
                                                      !
       ::::::::::::::    :::::::::::::::::            !
       : Televisore :    : Modulatore RF :            !
       : o Monitor  :----:               :------------!
       ::::::::::::::    :::::::::::::::::
```

Figura 6-1. Schema a blocchi del VIC

Figura 6-2. Veduta d'insieme del VIC aperto

6.5. UTILIZZO DELLA MEMORIA DEL VIC

Il microprocessore 6502 puo' indirizzare da 0 a 65535 usando la tecnica delle pagine di memoria. Vengono usati 2 byte per ogni indirizzo, il primo serve per indirizzare la pagina ed il secondo per reperire la locazione nella pagina. Ogni pagina contiene 256 locazione indirizzate da 0 a 255. Si hanno 256 pagine indirizzate da 0 a 255. Se si calcola 256*256 si trova 65536 che sono tutte le possibili locazioni di memoria per il nostro calcolatore, indirizzabili da 0 a 65535.

Gli indirizzi possibili non sono usati tutti se si ha un calcolatore nella sua prima configurazione e cioe' con 5K di RAM.

Vediamo come sono utilizzati i blocchi di memoria, sia RAM che ROM.

Indirizzi decimali		Indirizzi esadecimali
0	AREA DI LAVORO PER SISTEMA	0000
1023	OPERATIVO E BASIC RAM1K	03FF
1024	ESPANSIONE RAM	0400
4095	RAM3K	0FFF
4096	AREA RAM UTENTE DISPONIBILE	1000
7679	NELLA VERSIONE 5K RAM3.5K	1DFF
7680	MAPPA CARATTERI MEMORIA	1E00
8191	DI SCHERMO RAM0.5K	1FFF
8192	ESPANSIONE RAM/ROM	2000
16383	RAM/ROM 8K	3FFF
16384	ESPANSIONE RAM/ROM	4000
24575	RAM/ROM 8K	5FFF
24576	ESPANSIONE RAM/ROM	6000
32767	RAM/ROM 8K	7FFF
32768	IMMAGINE DEI CARATTERI	8000
36863	4 BLOCCHI DA 1K ROM4K	8FFF
36864	INDIRIZZI DI SISTEMA	9000
	ROM	
37136	INPUT/OUTPUT 0	9110
	RAM	
38400	MAPPA COLORI DEL VIDEO	9600
	RAM 0.5K	
38912	INPUT/OUTPUT 2	9800
	RAM	
39936	INPUT/OUTPUT 3	9C00
	RAM	
40960	ESPANSIONE ROM	A000
	ROM 8K	
49152	INTERPRETE BASIC	C000
	ROM 8K	
57344	SISTEMA OPERATIVO 8K	E000
	ROM	
65535		FFFF

Vediamo ora l'utilizzo dei bytes da 0 a 1023:

```
 Indirizzi                                          Indirizzi
 decimali                                           esadecimali
               ------------------------------------
      0      !    AREA DI LAVORO BASIC           !    0000
               ------------------------------------
    144      !    AREA DI LAVORO SISTEMA OP.     !    0090
               ------------------------------------
    256      !    AREA STACK BASIC E SIST. OP.!       0100
               ------------------------------------
    512      !    AREA LAVORO BASIC E SIST. OP.!      0200
               ------------------------------------
    828      !    BUFFER CASSETTA                  !  033C
   1023        ------------------------------------   0400
```

L'area stack e' usata con indirizzi decrescenti.

Vediamo ora l'utilizzo dei 3.5K utente (4096-7679):

```
 ----------------------------------
 !  PROGRAMMA BASIC           :   !
 !                            :   !
 !                            :   !
 !- - - - - - - - - - - - - - : - !
 !  VARIABILI E VARIABILI CON :   !
 !  INDICE                    --- !
 !- - - - - - - - - - - - - - - - !
 !                                !
 !                            --- !
 !- - - - - - - - - - - - - - : - !
 !  STRINGHE                  :   !
 !                            :   !
 ----------------------------------
```

il programma inizia al byte 4096 per indirizzi di bytes crescenti; subito sotto il programma vengono memorizzate le variabili semplici e poi quelle con indice, per le stringhe, sia semplici che con indice, in questa zona stanno solo i nomi, le dimensioni e i puntatori; il corpo delle stringhe sta nell'area a indirizzi piu' alti e viene memorizzato per indirizzi decrescenti.

Quando le stringhe crescono troppo si puo' avere il messaggio OUT OF MEMORY.

Vediamo l'utilizzo dei 4K per l'immagine dei caratteri (32868-36863):

. da 32768 a 33791 si hanno 1024 byte per i 128 possibili

caratteri del set grafico da far apparire sullo schermo (confrontate con Appendice B). Ogni carattere e' rappresentato in un blocco di 8 byte, cioe' si hanno a disposizione 8*8=64 bit per rappresentare un carattere. Si dice che il carattere e' rappresentato a punti in una matrice 8x8. Se moltiplichiamo 128x8 otteniamo 1024. Vediamo come e' rappresentato il carattere A:

```
0 0 0 1 1 0 0 0     e togliendo         1 1
0 0 1 0 0 1 0 0     gli zeri per      1     1
0 1 0 0 0 0 1 0     vedere meglio   1         1
0 1 1 1 1 1 1 0                     1 1 1 1 1 1
0 1 0 0 0 0 1 0                     1         1
0 1 0 0 0 0 1 0                     1         1
0 0 0 0 0 0 0 0
```

Il carattere A che corrisponde al codice 1 per la memoria di schermo si trova all'indirizzo:

32768 + 8 * codice e quindi 32768 + 8 * 1 = 32776

se andate a scrivere PRINT PEEK(32776) vedrete il numero 24 che in binario e':
00011000, se scrivete PRINT PEEK(32777) vedrete 36 che in binario e': 00100100, e cosi' via. Quindi nella mappa della memoria di schermo si trovano i codici dei caratteri, ma essi sono usati dal sistema come puntatori per andare a prelevare, dopo averli moltiplicati per 8 ed avere aggiunto l'indirizzo base, che in questo caso e' 32768, il disegno per punti del carattere. Questo disegno per punti e' quello che appare sul video. Il video ha quindi una risoluzione per punti di (22x8)x(23x8), cioe' di 176x184.

. da 32792 a 34815 si trovano i caratteri del set grafico in campo inverso. Se provate ad aggiungere 1024 all'indirizzo usato prima per vedere l'immagine di A ed usate ancora il comando PEEK vedrete una immagine nella quale gli zeri sono stati scambiati con gli uno.

. da 34816 a 35839 si trovano i caratteri del set maiuscolo/minuscolo.

. da 35840 a 36863 si trovano i caratteri del set maiuscolo/minuscolo in campo inverso.

Nella prima parte della memoria sono localizzati i puntatori per mezzo dei quali si puo' vedere dove si trovano le diverse zone in cui e' divisa la memoria utente; passiamo ad elencarli:

Indirizzi decimali byte basso	alto	Contenuto
43	44	indirizzo inizio programma
45	46	indirizzo inizio variabili
47	48	indirizzo inizio variabili con indice
49	50	indirizzo fine variabili con indice
51	52	indirizzo inizio stringhe (la prima caricata)
53	54	indirizzo fine stringhe
55	56	indirizzo fine memoria
65	66	indirizzo DATA

Di norma 43 e 44 contengono l'indirizzo 4096, dove inizia il programma.

Per calcolare gli indirizzi dovete procedere cosi':

```
PRINT(PEEK(44))*256 + PEEK(43)
```

e similmente per gli altri.

6.6. MEMORIZZAZIONE DELLE ISTRUZIONI E DEI DATI

Le linee di programma Basic sono memorizzate una dopo l'altra con il seguente formato:

. due bytes per il LINK, cioe' per puntare alla prossima linea, se i 2 bytes di LINK sono zero il programma termina; per calcolare l'indirizzo dove e' memorizzata la prossima linea si deve calcolare secondo byte moltiplicato 256 + primo byte;

. due byte per il numero di linea: il primo byte da' la parte meno significativa del numero linea ed il secondo la piu' significativa;

. segue la linea Basic e vengono usati i codici dell'Appendice J per le parole chiave ed i simboli ed i codici ASCII per i dati del programmatore;

. la linea termina con un byte a zero binario (tutti bit zero).

Se la linea di programma contiene spazi questi vengono conservati, con spreco di memoria. Non viene conservato lo spazio tra il numero di linea e il primo carattere della frase. In fase di LIST questo spazio viene aggiunto.

Il byte 4096 contiene sempre zero, quindi la prima linea di programma inizia in 4097.

Le variabili singole sono memorizzate cosi':

. per ogni variabile sono usati 2 bytes per il nome ed in essi e' contenuto anche il tipo della variabile:

	byte 1	byte 2
Interi	primo car. + 128	secondo car. + 128 o 128
Decimali	primo car.	secondo car. o 0
Stringhe	primo car.	secondo car. + 128 o 128

. seguono per ogni variabile 5 bytes ed essi vengono usati cosi':

	byte 3	byte 4	byte 5	byte 6	byte 7
Interi	valore attuale				
	256*HI	LO	0	0	0
Decimali	valore attuale del numero in floating point				
	esponente	mantissa			
Stringhe	contatore car.	puntatore al corpo della stringa			
		LO	HI	0	0

Nello schema LO significa byte con valore meno significativo e HI significa byte con valore piu' significativo.
Come vedete per le stringhe si ha solo il nome, il contatore dei caratteri ed il puntatore alla zona di memoria dove la stringa e' memorizzata.

Le variabili con indice sono invece memorizzate con la seguente testata:

byte 1 e 2	byte 3 e 4	byte 5	byte 6 e 7	byte 8 e 9
come per le var. singole nome	numero totale byte occupati	numero dimensioni	numero di elementi ultima dimensione	numero di elementi penultima dimensione

se la variabile ha due dimensioni sono occupati i 9 byte di cui sopra, se ha piu' dimensioni si deve aggiungere una coppia di byte per ogni dimensione.
Dopo la testata in dipendenza dal tipo si hanno:

- . 2 byte per ogni elemento se numeri interi;
- . 5 byte per ogni elemento se numeri decimali;
- . 3 byte per ogni elemento se stringhe e ogni terna contiene il numero dei caratteri e il puntatore.

come si puo' dedurre da quanto detto sopra, in un programma conviene definire come numeri interi solo le variabili con indice, perche' in tale caso si ha risparmio di memoria rispetto ai numeri decimali. Nel caso di variabili singole lo spazio usato e' il medesimo.

Per le variabili stringa, sia singole che con indice, il corpo della stringa viene allocato dinamicamente in memoria durante l'esecuzione del programma.

6.7. COME LAVORA IL SISTEMA

Il sistema puo' lavorare in due modi: stato sistema e stato programma. Al momento dell'accensione quando appare la scritta READY si e' nello stato sistema. Voi potete colloquiare con il VIC in BASIC.
Quando cominciate a scrivere una linea di programma e' attivo un programma chiamato "Editor" che vi consente di operare tramite il video per scrivere e modificare la frase. Quando poi premete il RETURN, la frase viene memorizzata,

andando a sostituirsi ad una eventuale frase gia' presente con lo stesso numero di linea, oppure prima della prima frase gia' esistente con numero di linea maggiore. La ricerca del posto dove memorizzare la frase viene fatta muovendosi con i 2 byte di LINK da una frase all'altra. Fino a quando non avete finito di scrivere il programma lavorate sempre con il calcolatore nello stato sistema.

Quando date il RUN al programma, il calcolatore passa allo stato programma e ritorna allo stato sistema se c'e' un errore, se voi premete STOP RUN, se premete RUN STOP e RESTORE o se il programma termina.

Durante l'esecuzione del programma e' attivo l'Interprete BASIC, il quale provvede all'esecuzione di ogni linea di programma. L'interprete si accorge della fine di una linea di programma dal byte a zero binario che chiude la linea.

6.8. IL REGISTRATORE A NASTRO

Se collegate il registratore a nastro al VIC potete conservare i vostri programmi ottenendo un buon risparmio di tempo.

Per memorizzare un programma si usa il comando SAVE, seguito o meno dal nome del programma. E' consigliabile usare sempre un nome per i programmi. Il registratore deve essere inattivo per evitare guai e il nastro deve essere posizionato al punto giusto.

Al comando SAVE il sistema risponde:

PRESS PLAY & RECORD ON TAPE

e voi dovete premere contemporaneamente i due tasti; il sistema risponde OK e poi SAVING. Quando la registrazione e' terminata vedrete che il nastro si ferma ed appare la scritta READY.

E' consigliabile verificare la bonta' della registrazione. Si deve riavvolgere il nastro e dare il comando VERIFY. A questo il sistema risponde:

PRESS PLAY

e dopo vedrete OK e SEARCHING e poi FOUND e VERIFYING e alla fine se tutto e' andato bene OK e READY. Se la verifica non ha dato buon esito vedrete ?VERIFY ERROR.

Per caricare in memoria un programma da nastro si usa il comando LOAD seguito dal nome del programma. Se il nome manca carica il primo programma presente su nastro. Il VIC risponde PRESS PLAY ON TAPE e a fine caricamento vedrete apparire READY. Se ci sono dei problemi di caricamento vedrete ?LOAD ERROR.

I dati sono gestiti a blocchi di 192 caratteri per volta, infatti il buffer per il nastro (parte di memoria dedicata all'ingresso e l'uscita) puo' contenete tanti caratteri. Il programmatore non deve comunque preoccuparsi di questo perche' pensa il sistema a gestire il buffer.

I programmi sono memorizzati su nastro esattamente come si trovano in memoria.

Il trattamento dei files di dati verra' svolto nel prossimo volume dedicato al VIC.

CAPITOLO 7

IMPARIAMO A PROGRAMMARE

7.1. I LINGUAGGI DI PROGRAMMAZIONE

Programmare un calcolatore in linguaggio macchina non e' una cosa semplice; per questa ragione sono stati messi a punto alcuni linguaggi simbolici di piu' facile apprendimento per l'uomo. Tali linguaggi non sono comprensibili per il calcolatore, ma lo diventano se esiste un programma che traduce da linguaggio simbolico a linguaggio macchina. Vi chiederete come si scrive il programma traduttore. Il primo si deve necessariamente scrivere in linguaggio macchina.

Il vantaggio di questa procedura e' che se il programma traduttore e' stato preparato bene esso puo' tradurre infiniti programmi scritti in linguaggio simbolico senza fare errori.

Ogni linguaggio simbolico ha le sue regole di grammatica e di sintassi e l'utente deve rispettarle. Il programma traduttore si basa infatti su quelle stesse regole e da una ben definita struttura di partenza arriva a produrre il programma in linguaggio macchina.

Esistono molti linguaggi simbolici di programmazione; possiamo ripartirli in due grandi gruppi:

- linguaggi compilativi o assemblativi;
- linguaggi interpretativi.

Al primo gruppo appartengono quei linguaggi che subiscono una traduzione prima di essere eseguiti sul calcolatore. Il programma scritto in linguaggio simbolico si chiama "Programma Origine", il programma tradotto si chiama "Programma oggetto" ed e' quest'ultimo che viene eseguito sul calcolatore.

Sono compilativi i linguaggi come il FORTRAN, il COBOL; essi vengono anche chiamati linguaggi ad alto livello perche' una istruzione di programma corrisponde a molte istruzioni in linguaggio macchina.

Sono invece assemblativi i linguaggi di tipo ASSEMBLER, linguaggi cioe' nei quali la maggior parte delle istruzioni, scritte in simbolico, da' luogo ad una sola istruzione in linguaggio macchina. Da cui il nome di linguaggi a basso livello.

Al secondo gruppo appartengono quei linguaggi che vengono

interpretati e quindi tradotti durante l'esecuzione del programma. Il BASIC e' un linguaggio di questo tipo. Il programma scritto in BASIC coesiste nella memoria del calcolatore con il programma INTERPRETE. Ogni frase del programma dell'utente viene interpretata ed eseguita subito, poi si passa alla frase successiva. Il fatto che l'interprete Basic stia in ROM ed il nostro programma in RAM non cambia la sostanza delle cose. RAM e ROM sono memorie che coesistono.

Il linguaggio BASIC si presta molto bene ad un primo approccio con la programmazione perche' non e' difficile, e' conversazionale, segnala subito gli errori di sintassi, consente di correggere gli errori in fase di prova del programma.

Voi potete imparare anche a programmare in linguaggio macchina sul VIC, ma sara' meglio rimandare questa parte a quando sarete diventati esperti programmatori Basic.

Un approccio sistematico alla programmazione dovrebbe partire dal linguaggio macchina ed arrivare ai linguaggi simbolici; seguire la strada inversa consente di ricavare subito qualche soddisfazione. Coloro che si appassionano potranno poi arrivare a tutto il resto.

Il vostro calcolatore ha residenti in ROM l'Interprete Basic ed il Sistema Operativo. Quest'ultimo e' un insieme di programmi che facilita l'uso del calcolatore. In seguito potra' essere interessante studiare come e' costruito il sistema operativo del VIC.

Vorrei concludere dicendovi che anche con il solo VIC potete farvi una buona cultura sui calcolatori elettronici; c'e' infatti la possibilita' di utilizzare altre periferiche. Se volete solo divertirvi un po' con il calcolatore e magari organizzarvi anche delle piccole gestioni di dati potete fermarvi al Basic. Se invece desiderate andare avanti,anche sul VIC ci sono tante cose da imparare.

7.2. IL PROGRAMMA

Un programma e' una sequenza di istruzioni che vengono impartite a qualcuno. Qualcuno puo' essere un uomo o una macchina che capisca le istruzioni.

Un programma per un calcolatore elettronico deve essere molto preciso perche' la macchina non ha fantasia, sa fare poche cose. Se si vogliono buoni risultati bisogna proprio dirgli tutto quello che deve fare.

Noi ora dobbiamo scrivere programmi in linguaggio Basic. E' necessario conoscere il linguaggio, avere capito un po' come funziona il calcolatore o almeno come esso reagisce ai nostri comandi e, cosa piu' difficile, sapere impostare i problemi che vogliamo programmare. Qualora un programma debba essere conservato e trasmesso ad altri, e' necessario che esso sia ben documentato, cioe' che sia accompagnato da annotazioni chiarificatrici sugli scopi e sul contenuto.

7.3. IL PROBLEMA

La prima cosa e' sapere chiaramente cosa si vuole fare; si puo' scrivere il testo del problema o averlo ben chiaro in mente. A mio avviso e' meglio scrivere cosa si vuole fare e rifletterci un po' sopra.

Si deve poi precisare quali sono i dati di ingresso su cui il futuro programma dovra' operare. Sarebbe bene cominciare a preparare un gruppo di dati che serviranno poi per collaudare il programma.

Si devono precisare i risultati che si vogliono ottenere e anche questa volta sarebbe bene preparare i risultati che si aspettano in conseguenza dei dati di ingresso gia' preparati.

Ogni programma consiste in una trasformazione di dati, quello che deve essere deciso e' il modo nel quale questa trasformazione deve avvenire. La procedura da svolgere deve essere scritta chiaramente; in generale si dovranno applicare uno o piu' algoritmi di calcolo. A questo proposito e' bene tener presente che alcune procedure, che sarebbe pazzesco pensare di realizzare in modo manuale, sono invece molto adatte ad essere svolte da un programma per calcolatore; questo dipende dalla altissima velocita' di calcolo del mezzo. Gli algoritmi di calcolo devono essere il piu' possibile schematizzati.

Quando il problema e' stato sufficientemente studiato ed esiste una traccia scritta di quello che si vuole fare si puo' passare alla stesura del programma per il calcolatore. Non e' una buona abitudine programmare stando seduti davanti alla tastiera, si deve preparare prima con calma il programma e scriverlo su un foglio di carta.

Terminato il programma viene il momento della prova. Non capita quasi mai che un programma non abbia errori. Alcuni errori sono banali e si limitano a errori di scrittura, altri sono sostanziali e dipendono da una errata impostazione logica del problema. Il sistema segnala solo gli errori di linguaggio e non gli errori di logica. Questi ultimi li deve trovare il programmatore riflettendo sui risultati ottenuti.

Se avete seguito con pazienza tutti i capitoli di questo

manuale ed avete riflettuto su ogni esempio avrete gia' assimilato alcune tecniche di programmazione, queste vi potranno essere utili nella stesura dei vostri programmi. Una cosa fondamentale e' l'esperienza e la si raggiunge cercando di scrivere e provare programmi, ma anche cercando di capire come altri hanno scritto programmi che funzionano. Ricordatevi che non e' molto importante scrivere un programma con poche istruzioni, ma e' molto importante scrivere un programma chiaro, tale cioe' che voi stessi dopo un po' di tempo, o altri possano facilmente capirlo.

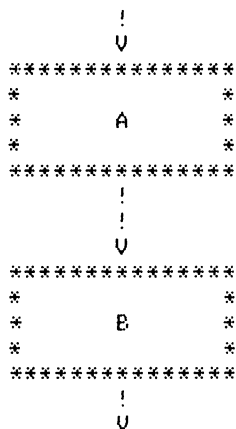
7.4. LE SITUAZIONI LOGICHE

Considerate un qualunque problema, se scrivete su un foglio di carta una dopo l'altra le operazioni che dovete svolgere vedrete che vi troverete in una delle seguenti situazioni:

.1) una operazione deve essere svolta dopo un altra;
.2) si deve operare una scelta sul cosa fare dopo a seconda del risultato ottenuto;
.3 una certa sequenza di operazioni deve essere ripetuta un certo numero di volte.

Ognuna di queste situazione ha un nome:

.1) sequenza;
.2) diramazione;
.3) iterazione (ciclo).

Si puo' mettere in evidenza ognuna di queste situazioni logiche servendosi di uno schema grafico, cosi':

SEQUENZA:
esegui B dopo avere eseguito A;

DIRAMAZIONE:
se P e' vero esegui A, altrimenti B;

ITERAZIONE:
esegui A fino a quando la situa=
zione P e' vera.

Si puo' anche pensare di descrivere queste 3 situazioni logiche a parole.

Possiamo concludere che la risoluzione di qualunque problema puo' essere ricondotta a una ripetizione di queste 3 situazioni, le quali hanno una caratteristica comune: esse hanno un solo punto di entrata e un solo punto di uscita.

In tutti i linguaggi di programmazione si trovano frasi che consentono di programmare facilmente queste strutture logiche. Ripensando agli esempi visti potete cercare di vedere dove si verifica una o l'altra situazione.

La prima, la sequenza, e' la piu' comune e si verifica sempre, fino a quando non intervengano le altre due. Le linee di programma Basic vengono svolte in sequenza.

La situazione di diramazione o scelta viene affrontata con la frase IF..THEN e l'abbiamo gia' incontrata varie volte.

Le iterazioni si programmano con il FOR...NEXT ed anche su questa parte avete gia' una piccola esperienza.

Si potrebbe osservare che nel BASIC si hanno alcune frasi molto potenti non facilmente riconducibili a queste 3 situazioni, tipo ON.. GOTO...; in realta' si potrebbe fare a meno di queste frasi e risolvere invece con una catena di IF..THEN.

7.5. LA SCHEMATIZZAZIONE

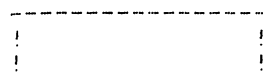
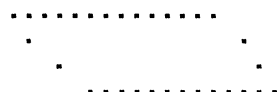
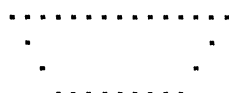
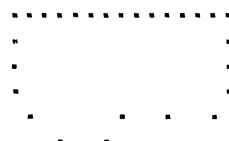
Per semplificare il lavoro di programmazione si deve imparare a schematizzare le procedure. La schematizzazione si puo' raggiungere in diversi modi.

Taluni possono trovarsi bene scrivendo i diversi passi come punti numerati, in tale modo sono possibili i richiami, cosi':

.1) faccio.......
.2) eseguo.......
.3) se......ritorno a 1)
.4) faccio.......
ecc..............

Da uno schema di questo tipo si ricava facilmente un programma.

Altri possono preferire una schematizzazione di tipo grafico e cioe' ricorrere alla tecnica dei diagrammi a blocchi.
Si usano un certo numero di simboli grafici recanti all'interno delle scritte che specificano le operazioni da compiere. I simboli stessi hanno un significato invalso nell'uso e nel paragrafo precedente ne abbiamo riportati alcuni.
Anche dai diagrammi a blocchi si ricavano agevolmente i programmi.

PRINCIPALI SIMBOLI USATI NEI DIAGRAMMI A BLOCCHI

```
 -----------------
 !               !          Elaborazione
 !               !
 -----------------


  . . . . . . . . . . . . .
   .             .          Ingresso/Uscita dati
     .             .
       . . . . . . . . . . . .


   . . . . . . . . . . . . .
    .            .          Visualizzazione
     .         .
      . . . . . . . .


      -----
      !   !                 Operazioni ausiliarie
      -----


   . . . . . . . . . . . . . .
   .                        .   Stampa su carta
   .                        .
   .                        .
    .     .     .     .
      .     .


         *
     *       *              Scelta (diramazione)
  *             *
     *       *
         *


    -----------------
    ! !           ! !       Chiamata sottoprogrammi
    ! !           ! !
    ! !           ! !
    -----------------
```

La schematizzazione trattata fino ad ora riguarda il modo nel quale preparare un lavoro di programmazione tenendo l'attenzione puntata sul problema da risolvere.

L'esperienza nel lavoro di programmazione consente poi di

adottare delle tecniche di programmazione che consentono di semplificare i programmi. Possiamo suggerire di abituarsi a scrivere programmi semplici e corti. Corti nel senso che e' meglio spezzare un problema nelle sue componenti piu' semplici e scrivere tanti sottoprogrammi di poche istruzioni, ciascuno avente uno scopo ben determinato. I linguaggi di programmazione consentono di richiamare i pezzi di programma con la tecnica del richiamo dei sottoprogrammi (GOSUB/RETURN). Un programma complesso si riduce cosi' in una serie di chiamate a sottoprogrammi.

Inoltre se i diversi sottoprogrammi vengono conservati e sono ben documentati, essi possono essere utilizzati piu' volte da programmi diversi. Per evitare il fastidio della rinumerazione basta assegnare ai sottoprogrammi numeri di linea bassi e decidere di far partire i programmi da numeri di linea che iniziano, per esempio, da 1000.

7.6. IL BASIC DEL VIC

Un programma BASIC e' formato da linee numerate progressivamente. I numeri di linea possono partire da 0 e arrivare a 63999.

Il BASIC puo' lavorare in due modi: modo immediato e modo differito. Le frasi senza numero di linea sono in modo immediato e vengono eseguite appena sono accettate dal calcolatore alla pressione del tasto RETURN. Le frasi precedute dal numero di linea vengono memorizzate alla pressione del tasto RETURN e sono eseguite in modo differito quando si da' il comando RUN di esecuzione programma.

Il programmatore puo' anche scrivere le frasi di programma non ordinate per numero di linea crescente; il sistema le memorizza in ordine di numero di linea crescente. Una frase recante lo stesso numero di una linea gia' esistente si sovrappone alla precedente cancellandola.

E' buona norma non usare numeri di linea consecutivi, ma lasciare un intervallo numerico tra una linea e l'altra onde consentire correzioni o aggiunte future.

Prima di cominciare a scrivere un programma e' bene cancellare la memoria con il comando NEW.

Una linea di programma puo' arrivare fino a 88 caratteri cioe' 4 linee di video, puo' contenere piu' di una istruzione purche' si usi il separatore due punti. Per abolire una linea di programma basta scrivere il numero di linea e premere RETURN.

Il programma viene svolto eseguendo una linea dopo l'altra per numeri di linea crescenti fino a quando viene incontrata una istruzione che interrompe la sequenza e rimanda ad un

ben definito numero di linea.

Nella Appendice E trovate tutte le istruzioni del linguaggio.

Seguono alcuni consigli per risparmiare memoria e rendere piu' veloci i programmi:

. scrivere linee di programma con piu' di una istruzione;
. non mettere inutili spazi nelle istruzioni;
. usare pochi REM e scrivere la documentazione a parte;
. usare variabili invece di costanti, cioe' definire le costanti con un nome e poi richiamarle con il nome;
. non scrivere END alla fine del programma;
. riutilizzare le stesse variabili, se possibile, in punti diversi di un programma;
. fare ampio uso della tecnica dei sottoprogrammi e porre i sottoprogrammi all'inizio del programma;
. usare la posizione zero delle variabili con indice;
. porre i DATA in fondo al programma;
. usare variabili intere con indice (%) per gli insiemi di numeri interi;
. scrivere i NEXT senza citare la variabile;
. definire prima le variabili che si usano piu' di frequente.

7.7. COSTRUIAMO DUE PROGRAMMI

Primo problema:

LANCIANDO N VOLTE UN DADO SI VUOLE SAPERE QUANTE VOLTE ESCE OGNI NUMERO.

Il numero dei lanci N puo' variare ogni volta e viene chiesto come INPUT dal programma. Le facce del dado sono 6 e recano i numeri da 1 a 6. Si devono organizzare 6 variabili per contare le frequenze dei risultati dei lanci. Il programma deve lavorare ciclicamente e il ciclo si deve ripetere N volte. Si deve stabilire come ottenere il lancio; si presta bene la funzione RND se il numero pseudo random ottenuto si moltiplica per 6 , si rende intero e si aggiunge 1. Si deve stabilire come comunicare i risultati. Si puo' molto semplicemente evidenziare una tabellina sul video, dove ad ogni numero si fa corrispondere la frequenza ottenuta.

Procediamo all'analisi discorsiva per punti:

.1) Richiesta e memorizzazione del numero dei lanci N
.2) Inizio di un ciclo di programma governato dalla variabile (contatore) K che parte da 1 e arriva ad N con

STEP=1
.3) Estrazione di un numero C a caso e sua riduzione all'intervallo 1-6.
.4) Il numero C serve come indice per la variabile con indice che rappresenta le 6 caselle per calcolare le frequenze (definite con DIM F(6)). Viene aggiunto 1 a F(C).
.5) Se il ciclo non e' esaurito si ritorna al punto 3), se e' esaurito si prosegue.
.6) Si stampano una frase di intestazione e il numero dei lanci effettuati.
.7) Si stampa su una riga DADO e FREQUENZA.
.8) Si stampa con un ciclo il numero della faccia del dado e la frequenza riscontrata.

La codifica risultante e':

```
 10 REM CALCOLO FREQUENZE NEL LANCIO
 20 REM DI UN DADO N VOLTE
 30 DIM F(6)
 40 INPUT"QUANTE VOLTE";N
 50 FOR K=1 TO N
 60 C=RND(0)
 70 C=INT(C*6)+1
 80 F(C)=F(C)+1
 90 NEXTK
100 PRINT"IN ";N;" LANCI SI E' OTTENUTO:"
110 PRINT "DADO","FREQUENZA"
120 FOR K=1 TO 6:PRINT K,F(K):NEXTK
```

Come si fa a sapere se il programma lavora bene? Una verifica puo' essere quella di sommare le frequenze e vedere se il totale e' uguale a N.
Un'altra verifica puo' essere quella di mettere uno STOP tra le linee 70 e 80 e vedere il valore di C con una PRINT in immediato. Si sarebbe potuto introdurre la PRINT di C nel programma, ma per poter leggere si sarebbe dovuto rallentare lo scrolling.

Secondo problema:

SCRIVERE UN SOTTOPROGRAMMA CHE METTA IN ORDINE CRESCENTE N NUMERI MEMORIZZATI IN NU(K).

Sottoprogramma vuol dire che alla fine logica delle istruzioni dobbiamo scrivere RETURN. Dato che si tratta di un sottoprogramma, esso lavorera' sempre insieme ad un programma; sara' il programma a riempire di N numeri NU(K). Nel sottoprogramma non deve esserci la DIM di NU(N) e non deve esserci la lettura di N. Esso deve solo mettere in ordine crescente gli N numeri che trova in NU(K).

Prima di procedere all'analisi per punti, esponiamo l'algoritmo di ordinamento scelto. Si tratta del ben noto ordinamento a bolle. Questo nome deriva dal fatto che come le bolle salgono verso l'alto, cosi' nel nostro insieme di numeri gli elementi ordinati si cominciano a disporre a partire dal basso.

Si confrontano i numeri a coppie, partendo dagli indici piu' bassi e operando degli scambi se i numeri non sono in ordine. Prima di ogni ispezione si pone a zero un Flag di controllo che viene messo a 1 se si e' operato uno scambio. Se alla fine di una ispezione il Flag e' rimasto a zero questo significa che gli elementi sono in ordine. Inoltre ad ogni ispezione si puo' tralasciare di controllare l'ultimo elemento perche' esso sara' diventato il maggiore tra quelli confrontati. Non si conosce a priori quante ispezioni si fanno, sicuramente non piu' di N-1.

. 1) Si pone la variabile C=N.
. 2) Si inizia un ciclo per contare N-1 ispezioni.
. 3) Si pone il Flag F=0.
. 4) Si inizia un ciclo per J che va da 1 a C-1.
. 5) Si confronta NU(J) con NU(J+1)
. 6) Se NU(J)<=NU(J+1) si va a 8).
. 7) Se no si scambiano tra loro i due elementi servendosi di una variabile di transito CO e si pone F=1.
. 8) Se J ha superato C-1 si prosegue, se no si torna a 5).
. 9) Se F=0 l'ordinamento e' terminato, si pone K=N-1 e si va a 11).
.10) Si diminuisce C di 1 e si torna a 3).
.11) Se il ciclo di N-1 ispezioni e' terminato si esce, altrimenti si torna a 3).

La codifica e':

```
 10 REM SOTTOPROGRAMMA ORDINAMENTO
 20 C=N
 30 FOR K=1 TO N-1
 40 F=0
 50 FOR J=1 TO C-1
 60 IF NU(J)<=NU(J+1)THEN 80
 70 CO=NU(J):NU(J)=NU(J+1):NU(J+1)=CO:F=1
 80 NEXTJ
 90 IF F=0 THEN K=N-1:GOTO110
100 C=C-1:GOTO 40
110 NEXTK
120 RETURN
```

Per provare questo sottoprogramma dovete scrivere un

programma che chieda N, dimensioni NU(N), incorpori il sottoprogramma ed abbia una istruzione GOSUB 10. Dopo il GOSUB dovrete scrivere le istruzioni per stampare i numeri ordinati e poterli verificare.

APPENDICE A

IL CODICE ASCII

Nella tabella che segue vengono mostrati:

. nella prima colonna i caratteri del Set Standard (Set Grafico);

. nella seconda colonna i caratteri del Set Alternativo (Set Maiuscolo/Minuscolo);

. nella terza colonna il Codice ASCII corrispondente.

Al momento dell'accensione del VIC e' attivo il Set di caratteri Standard e questo corrisponde al fatto che il byte di indirizzo 36869 contiene il numero 240. Il byte di indirizzo 36869 viene analizzato ed utilizzato dal software di gestione del video per la scelta del set di caratteri.

Se il byte di indirizzo 36869 contiene 242 diventa attivo il Set Alternativo di caratteri.

Per passare da un set di caratteri all'altro si puo' procedere in diversi modi che passiamo ad elencare:

.1) Premere contemporaneamente i due tasti COMMODORE e SHIFT. La pressione contemporanea di questi due tasti modifica il contenuto del byte 36869 facendolo passare da 240 a 242 e viceversa.

.2) Scrivere:

```
POKE 36869,240    oppure    POKE 36869,242
```

nel primo caso viene attivato il set standard, nel secondo il set alternativo.

.3) Scrivere cosi':

PRINT CHR$(14) per passare al set alternativo;

PRINT CHR$(142) per passare al set standard.

Se si usa la funzione CHR$(X), dove X puo' essere un numero compreso tra 0 e 255, si ottengono tutti i caratteri stampabili e inoltre i caratteri che rappresentano dei

comandi come CHR$(14) che fa passare al set alternativo, CHR$(28) che fa passare al colore rosso, ecc..

Nella tabella i caratteri che rappresentano comandi sono disegnati come tasti su sfondo nero. Come si puo' osservare, in ambedue i set di caratteri, a diversi valori di X corrispondono gli stessi caratteri stampabili. Inoltre ad alcuni valori di X non corrispondono caratteri stampabili.

La funzione CHR$(X) viene interpretata ed eseguita dal BASIC, ma X non e' il numero che viene realmente scritto nella memoria di schermo per ottenere i caratteri. Se consultate l'Appendice B potete trovare i valori che vengono trovati nella memoria di schermo per ogni carattere. Nella stessa appendice B e' esposto il calcolo per passare dal codice ASCII al codice usato per la memoria di schermo.
La funzione inversa della CHR$(X) e' la ASC("carattere") che fornisce il codice numerico corrispondente ad un determinato carattere battuto sulla tastiera o alla variabile stringa che contiene il carattere.

Questa tabella e' molto utili per quei caratteri comando che non si possono includere tra apici, come, per esempio, SHIFT e COMMODORE.

Alle pagine 18 e 19 e' riportato un programma per provare tutti i codici stampabili.

Segue la tabella dei codici ASCII.

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		0
		1
		2
		3
		4
WHT	WHT	5
		6
		7
		8
		9
		10
		11
		12
RETURN	RETURN	13
SWITCH TO LOWER CASE	SWITCH TO LOWER CASE	14
		15
		16
CRSR ⇓	CRSR ⇓	17
RVS ON	RVS ON	18
CLR HOME	CLR HOME	19
INST DEL	INST DEL	20
		21
		22
		23

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		24
		25
		26
		27
RED	RED	28
CRSR ⇒	CRSR ⇒	29
GRN	GRN	30
BLU	BLU	31
SPACE	SPACE	32
!	!	33
"	"	34
#	#	35
$	$	36
%	%	37
&	&	38
'	'	39
(	(	40
)	)	41
*	*	42
+	+	43
,	,	44
-	-	45
		46
/	/	47

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
0	0	48
1	1	49
2	2	50
3	3	51
4	4	52
5	5	53
6	6	54
7	7	55
8	8	56
9	9	57
:	:	58
;	;	59
<	<	60
=	=	61
>	>	62
?	?	63
@	@	64
A	a	65
B	b	66
C	c	67
D	d	68
E	e	69
F	f	70
G	g	71

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
H	h	72
I	i	73
J	j	74
K	k	75
L	l	76
M	m	77
N	n	78
O	o	79
P	p	80
Q	q	81
R	r	82
S	s	83
T	t	84
U	u	85
V	v	86
W	w	87
X	x	88
Y	y	89
Z	z	90
[	[	91
£	£	92
]	]	93
↑	↑	94
←	←	95

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		96
♠	A	97
	B	98
	C	99
	D	100
	E	101
	F	102
	G	103
	H	104
	I	105
	J	106
	K	107
	L	108
╲	M	109
╱	N	110
	O	111
	P	112
●	Q	113
	R	114
♥	S	115
	T	116
	U	117
╳	V	118
○	W	119

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
♣	X	120
	Y	121
◆	Z	122
		123
		124
		125
π		126
		127
		128
		129
		130
		131
		132
f1	f1	133
f3	f3	134
f5	f5	135
f7	f7	136
f2	f2	137
f4	f4	138
f6	f6	139
f8	f8	140
SHIFT RETURN	SHIFT RETURN	141
SWITCH TO UPPER CASE	SWITCH TO UPPER CASE	142
		143

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
BLK	BLK	144
↑ CRSR	↑ CRSR	145
RVS OFF	RVS OFF	146
CLR HOME	CLR HOME	147
INST DEL	INST DEL	148
		149
		150
		151
		152
		153
		154
		155
PUR	PUR	156
← CRSR	← CRSR	157
YEL	YEL	158
CYN	CYN	159
SPACE	SPACE	160
▌	▌	161
▄	▄	162
▔	▔	163
▁	▁	164
▏	▏	165
▒	▒	166
▕	▕	167

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		168
		169
		170
		171
		172
		173
		174
		175
		176
		177
		178
		179
		180
		181
		182
		183
		184
		185
	√	186
		187
		188
		189
		190
		191

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		192
♠	A	193
	B	194
	C	195
	D	196
	E	197
	F	198
	G	199
	H	200
	I	201
	J	202
	K	203
	L	204
	M	205
	N	206
	O	207
	P	208
●	Q	209
	R	210
♥	S	211
	T	212
	U	213
	V	214
	W	215

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
	X	216
	Y	217
	Z	218
		219
		220
		221
π		222
		223
		224
		225
		226
		227
		228
		229
		230
		231
		232
		233
		234
		235
		236
		237
		238
		239

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE ASCII
		240
		241
		242
		243
		244
		245
		246
		247
		248
		249
		250
		251
		252
		253
		254
π		255

APPENDICE B

IL CODICE PER LA MEMORIA DI SCHERMO

In questa tabella sono riportati i codici numerici che si trovano nella mappa dei caratteri della memoria di schermo. La mappa dei caratteri e' formata da 506 bytes di indirizzo da 7680 a 8185.

Se si legge con la funzione PEEK il contenuto dei bytes della mappa dei caratteri si ritrovano questi valori numerici. Se si vuole adoperare il comando POKE per scrivere direttamente nella mappa dei caratteri si devono adoperare questi codici e non quelli ASCII della Appendice A.

Nella tabella sono riportati i codici appartenenti ai due set di caratteri. Tali codici vanno da 0 a 127. Se si aggiunge 128 ad un codice si ottiene lo stesso carattere in campo inverso. I caratteri stampabili sono quindi 128. Se osservate bene i due set di caratteri vi accorgerete che il set alternativo ha meno caratteri grafici, dato che ha al posto di un gruppo di caratteri grafici le lettere maiuscole, pero' ha 4 caratteri grafici che non sono compresi nel primo set, quelli che corrispondono ai codici 94,95,105 e 122.

I due set di caratteri non sono mescolabili, si puo' usare l'uno o l'altro. Nella Appendice A si elencano tutti i modi per selezionare il set di caratteri desiderato.

La relazione matematica che lega i codici dell'appendice B con i codici ASCII e' la seguente, chiamando A il codice ASCII e D il codice della memoria di schermo (Display Code):

. se 32<=A<= 63 allora 32<=D<= 63 quindi D=A
. se 64<=A<= 95 allora 0<=D<= 31 quindi D=A-64
. se 96<=A<=127 allora 64<=D<= 95 quindi D=A-32
. se 160<=A<=191 allora 96<=D<=127 quindi D=A-64.

Segue la tabella dei codici per i caratteri della memoria di schermo.

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
@	@	0	128
A	a	1	129
B	b	2	130
C	c	3	131
D	d	4	132
E	e	5	133
F	f	6	134
G	g	7	135
H	h	8	136
I	i	9	137
J	j	10	138
K	k	11	139
L	l	12	140
M	m	13	141
N	n	14	142
O	o	15	143
P	p	16	144
Q	q	17	145
R	r	18	146
S	s	19	147
T	t	20	148
U	u	21	149
V	v	22	150
W	w	23	151

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
X	x	24	152
Y	y	25	153
Z	z	26	154
[	[	27	155
£	£	28	156
]	]	29	157
↑	↑	30	158
←	←	31	159
SPACE	SPACE	32	160
!	!	33	161
"	"	34	162
#	#	35	163
$	$	36	164
%	%	37	165
&	&	38	166
'	'	39	167
(	(	40	168
)	)	41	169
*	*	42	170
+	+	43	171
,	,	44	172
-	-	45	173
	.	46	174
/	/	47	175

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
0	0	48	176
1	1	49	177
2	2	50	178
3	3	51	179
4	4	52	180
5	5	53	181
6	6	54	182
7	7	55	183
8	8	56	184
9	9	57	185
		58	186
;	;	59	187
<	<	60	188
=	=	61	189
>	>	62	190
?	?	63	191
─	─	64	192
♠	A	65	193
│	B	66	194
─	C	67	195
▔	D	68	196
▔	E	69	197
▁	F	70	198
▏	G	71	199

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
	H	72	200
	I	73	201
	J	74	202
	K	75	203
	L	76	204
	M	77	205
	N	78	206
	O	79	207
	P	80	208
	Q	81	209
	R	82	210
	S	83	211
	T	84	212
	U	85	213
	V	86	214
	W	87	215
	X	88	216
	Y	89	217
	Z	90	218
		91	219
		92	220
		93	221
π		94	222
		95	223

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
SPACE	SPACE	96	224
		97	225
		98	226
		99	227
		100	228
		101	229
		102	230
		103	231
		104	232
		105	233
		106	234
		107	235
		108	236
		109	237
		110	238
		111	239
		112	240
		113	241
		114	242
		115	243
		116	244
		117	245
		118	246
		119	247

CARATTERI STANDARD	CARATTERI ALTERNATIVI	CODICE MEMORIA DI SCHERMO	
		NORMALE	CAMPO INVERSO
		120	248
		121	249
	√	122	250
		123	251
		124	252
		125	253
		126	254
		127	255

APPENDICE C

LE MAPPE DELLO SCHERMO

Riportiamo in questa appendice le due mappe della memoria di schermo, quella per i caratteri e quella per i colori. Esse possono essere utili per preparare grafica a colore. La posizione di ogni quadratino si ottiene sommando il numero della riga a quello della colonna. Gli interessati potranno ricavarne delle copie ed utilizzarle per preparare i loro lavori.

Ricordiamo che i numeri colore da usare per la mappa dei colori sono uguali al numero scritto sul corrispondente tasto - 1. I codici numerici per la mappa dei caratteri si ricavano dalla Appendice B.

Se al VIC viene aggiunta memoria cambiano gli indirizzi di riferimento delle due mappe. Per questa ragione e' meglio scrivere i programmi ponendo in due variabili gli indirizzi di inizio delle mappe e usare le variabili. In caso di modifica della memoria basta cambiare queste due variabili e tutto il resto va ancora bene.

Seguono le due mappe.

MAPPA DEI CARATTERI

MAPPA DEI COLORI

APPENDICE D

I MESSAGGI DI ERRORE

Il sistema segnala gli errori con dei messaggi, segue l'elenco di questi con le spiegazioni sul loro significato.

. BAD DATA - Il programma era in attesa di ricevere un input numerico da una periferica e sono arrivati caratteri invalidi.

. BAD SUBSCRIPT - Nel programma e' stata definita una variabile con indice con un definito numero di dimensioni e con definite dimensioni e una istruzione si riferisce ad una variabile con lo stesso nome, ma non corrispondente alla definizione. Esempio:

```
DIM X(2,3)         variabile X con 2 dimensioni, avente 3
                   righe e 4 colonne

X(1,1,1)=8         riferimento ad un elemento X avente 3
                   dimensioni invece di due
?BAD SUBSCRIPT ERROR  messaggio di errore

X(5,3)=8           riferimento ad un elemento X avente 2
                   dimensioni, ma la prima fuori dalla
                   definizione
?BAD SUBSCRIPT ERROR messaggio di errore
```

. CAN'T CONTINUE - e' stato dato un comando CONT, ma il programma non puo' proseguire per una delle seguenti ragioni:

1) non esiste un programma;
2) si e' appena scritta una nuova linea di programma;
3) il programma non ha ricevuto il comando di RUN;
4) e' appena stato segnalato un errore.

. DEVICE NOT PRESENT - la periferica richiesta per una operazione del tipo: OPEN, CLOSE, CMD, INPUT#, GET#, PRINT#, non e' presente.

. DIVISION BY ZERO - si e' tentato di dividere per zero e questo non e' consentito.

. EXTRA IGNORED - si sono scritti piu' dati di quanti richiesti in risposta ad un comando INPUT. Vengono accettati tanti dati quante sono le variabili presenti nel comando e

gli altri vanno persi, ma il programma continua.

. FILE NOT FOUND - non si trova il file sulla periferica: per il nastro si incontra una segnalazione di END OF TAPE dopo aver fatto scorrere il nastro, per il disco non si trova il nome del file nell'indice.

. FILE NOT OPEN - si e' usato un comando come: CLOSE, CMD, INPUT#, GET#, PRINT# per un file che non e' stato aperto.

. FILE OPEN - si e' tentato di aprire un file che e' gia' stato aperto.

. FORMULA TOO COMPLEX - si e' usata una formula di calcolo troppo complicata ed il sistema non riesce a calcolarla. Si deve spezzare il calcolo in due parti e calcolarle separatamente.

. ILLEGAL DIRECT - si e' tentato di usare in modo immediato o il comando INPUT o il comando GET o il comando DEF di definizione di una funzione e questo non e' consentito.

. ILLEGAL QUANTITY - si e' tentato di usare una funzione con un argomento fuori dai limiti consentiti. Questo puo' avvenire nei seguenti casi:

1) gli indici di una variabile con indice sono fuori dall'intervallo 0-32767, per esempio si e' usato un numero negativo;

2) per la funzione LOG si e' usato un argomento negativo o nullo;

3) per la funzione SQR si e' usato un argomento negativo;

4) A elevato B, dove A < 0 e B non intero;

5) si usa il comando USR, ma il programma in linguaggio macchina non e' presente in memoria;

6) si usano le funzioni di stringa MID$, LEFT$, RIGHT$ con parametri fuori dall'intervallo 1-255;

7) si usa il comando ON...GOTO con indici non validi;

8) si usano i comandi: PEEK, POKE, WAIT, SYS con parametri fuori dall'intervallo 0-65535;

8) si usano i comandi: WAIT, POKE, TAB, SPC con parametri fuori dall'intervallo 0-255.

. LOAD - si sta caricando un programma da nastro e si sono trovati piu' di 31 errori nel primo blocco, oppure ci sono errori in ambedue i blocchi.

. NEXT WITHOUT FOR - si puo' avere un errore nel concatenamento dei FOR/NEXT, oppure la variabile citata in NEXT non corrisponde a quella del FOR.

. NOT INPUT FILE - se un file e' stato aperto per scrivere non si puo' fare una operazione di INPUT.

. NOT OUTPUT FILE - si tenta di scrivere su un file che e' stato aperto per INPUT, oppure che e' solo di INPUT come la tastiera.

. OUT OF DATA - si tenta di leggere dall'interno del programma con un READ piu' dati di quanti ne sono stati forniti con le frasi DATA; a volte si ha questo errore premendo il RETURN dopo il READY. del video, che viene interpretato come READ Y.

. OUT OF MEMORY - puo' essere determinato dalle seguenti cause:

1) si sta scrivendo un programma troppo lungo;

2) si sta facendo girare un programma e le variabili che si creano occupano troppa memoria; le stringhe occupano spazio di memoria che varia in dipendenza della loro lunghezza;

3) nel programma il numero di FOR/NEXT o di GOSUB/RETURN supera la capacita' dell'area stack.

Per decidere se dipende dall'ultima causa 3), basta scrivere:

?FRE(0) e vedere se la risposta e' un numero diverso da zero.

. OVERFLOW - si sono eseguiti dei calcoli che hanno dato risultati troppo grandi rispetto alle possibilita' del calcolatore.

. REDIM'D ARRAY - le variabili con indice possono essere dimensionate una volta sola in un programma.

. REDO FROM START - si e' risposto ad una richiesta di INPUT numerico con caratteri non validi. Dopo il messaggio compare il punto interrogativo come nuova richiesta di INPUT ed il programma prosegue se si risponde in modo corretto.

. RETURN WITHOUT GOSUB - si e' incontrato un RETURN, ma prima non c'e' stato un GOSUB.

. STRING TOO LONG - si e' cercato di concatenare delle stringhe e la stringa risultante sarebbe piu' lunga di 255 caratteri.

. SYNTAX - si e' scritto un comando con qualche errore ed il VIC non lo riconosce.

. TYPE MISMATCH - si e' usato un tipo di dato errato, numero al posto di stringa o viceversa.

. UNDEF'D FUNCTION - si fa riferimento ad una funzione che non e' stata definita precedentemente.

. UNDEF'D STATEMENT - si e' usato un comando come: GOSUB, GOTO, THEN con riferimento ad un numero di linea inesistente.

. VERIFY - ci sono delle differenze nella comparazione tra il contenuto della memoria ed un file su disco o nastro.

Quando si ha un messaggio di errore il programma si ferma, le variabili mantengono il loro valore e possono essere analizzate con dei comandi in modo immediato, i riferimenti di programma nella stack area vengono persi (GOSUB e FOR) e quindi non si puo' proseguire nell'esecuzione. Si puo' pero ripartire con un GOSUB ad una linea opportuna, ed in tale modo non si perdono i valori delle variabili gia' calcolate. Oppure si puo' ripartire con RUN.

Gli unici errori che non fermano il programma sono:

REDO FROM START
EXTRA IGNORED.

Tra i messaggi riportati i seguenti sono errori segnalati dal Sistema Operativo:

BAD DATA
DEVICE NOT PRESENT
FILE NOT FOUND
FILE NOT OPEN
FILE OPEN
LOAD
NOT INPUT FILE
NOT OUTPUT FILE
VERIFY

tutti gli altri sono errori segnalati dall'interprete BASIC.

APPENDICE E

IL BASIC DEL VIC

LE VARIABILI

Sono disponibili 3 tipi di variabili:

- numeriche;
- numeriche intere;
- stringhe.

Le VARIABILI NUMERICHE, che sono anche chiamate floating point, possono avere valore assoluto circa compreso tra 10 elevato a -38 e 10 elevato a +38 e circa 9 cifre di precisione. I numeri vengono stampati in notazione decimale fino a quando sono compresi in valore assoluto tra 0.01 e 999999999; fuori da questo intervallo vengono stampati in notazione scientifica, con una cifra intera e 8 decimali seguiti dalla lettera E e dall'esponente con segno. In questo caso il valore del numero si ottiene moltiplicando il numero decimale esposto per 10 elevato all'esponente che appare dopo la lettera E. Per esempio il numero 2233445566 viene scritto: 2.2334556E+10.

Le VARIABILI NUMERICHE INTERE possono avere valore compreso tra -32768 e +32767.

Le VARIABILI STRINGA possono contenere qualunque carattere. In fase di INPUT possono contenere al massimo 88 caratteri. Se si opera sulle stringhe con le operazioni consentite si possono avere stringhe lunghe fino a 255 caratteri.
L'assegnazione dello spazio di memoria alle stringhe avviene in modo dinamico durante l'esecuzione del programma. Nella fase iniziale vengono solo predisposti gli spazi per il nome e per la definizione della stringa.

I NOMI DELLE VARIABILI possono essere formati da quanti caratteri si vuole, purche' il primo sia una lettera e gli altri siano lettere o cifre, essi devono terminare con % se si riferiscono a variabili intere e con $ se si riferiscono a stringhe. Il sistema riconosce le variabili dai primi due caratteri del nome, per cui e' abbastanza inutile usare nomi lunghi.

Le VARIABILI CON INDICE sono insiemi di variabili aventi un nome comune; per distinguere i singoli elementi

dell'insieme si fa seguire al nome della variabile una coppia di parentesi che contiene un indice. L'indice puo' essere un numero o una variabile numerica. Queste variabili vengono accettate dal sistema in due modi:

. mediante una frase di definizione DIM;
. senza frase di definizione, ed in questo caso ogni dimensione vale 10. Si ha questo dimensionamento implicito la prima volta che il programma chiama la variabile con indice.

Esempio:

```
10 DIM A(4),B$(3,4)
```

definisce una variabile numerica A ad una dimensione formata da 5 elementi, infatti il valore dell'indice parte da 0, ed una variabile stringa a due dimensioni formata da 4 x 5 = 20 elementi, che possono essere pensati organizzati in forma tabellare, come una tabella di 4 righe con 5 colonne ciascuna.

Il numero delle dimensioni puo' teoricamente essere 32767, ma evidentemente esso viene limitato in conseguenza della memoria disponibile.

Si puo' usare lo stesso nome per due variabili, una con indice e l'altra no:

A$ e A$(K) sono considerate due variabili diverse.

Il sistema usa le 3 variabili TI,TI$ e ST per suoi scopi e questi nomi non possono essere usati dal programmatore. TI parte da zero al momento dell'accensione del VIC, viene incrementata di 1 ogni sessantesimo di secondo, viene modificata se si modifica TI$. TI$ contiene una stringa di 6 caratteri che viene continuamente aggiornata in base al valore di TI e che contiene HHMMSS (ore-minuti-secondi). TI$ puo' essere modificata dall'utente; al momento dell'accensione parte da "000000".

ST contiene delle informazioni circa l'esito delle operazioni con le periferiche.

LE COSTANTI

Si hanno tanti tipi di costanti quanti sono i tipi di variabili e con le stesse caratteristiche. Le costanti stringa si possono, e in certi casi si devono, racchiudere tra apici. La stringa nulla e' quella che si ottiene scrivendo "" senza spazio ed e' diversa dalla stringa che si ottiene scrivendo uno spazio tra gli apici.

GLI OPERATORI

Si hanno 4 tipi di operatori:

- aritmetici;
- relazionali;
- logici;
- funzionali.

Nelle espressioni possono essere usati tutti i tipi di operatori, se gli operandi sono numerici, solo alcuni se gli operandi sono stringhe. In una espressione aritmetica si possono usare tutti gli operatori che producono risultati numerici. Per le stringhe si possono usare gli operatori relazionali e logici e l'operatore aritmetico + che serve a concatenare tra loro due stringhe, cioe' a scriverle una vicino all'altra. La stringa risultato di una operazione di concatenamento non puo' superare i 255 caratteri.

Nel linguaggio BASIC esistono delle funzioni gia' intrinsicamente definite e quando esse vengono usate in espressioni danno luogo ad un valore corrispondente; inoltre l'utente puo' definire delle funzioni con la frase DEF FN. Se in una espressione e' presente una funzione essa viene calcolata con precedenza su tutto il resto.

Le espressioni vengono calcolate da sinistra a destra dando la precedenza alle operazioni racchiuse in parentesi e rispettando le priorita' degli operatori. Le operazioni con priorita' maggiore vengono eseguite per prime.

Si riporta la tabella degli operatori, omettendo le funzioni:

TIPO	PRIORITA'	OPERATORE	COMMENTI
	9	()	parentesi che racchiudono espressioni
ARITMETICI	8	freccia su	elevamento a potenza
	7	-	segno numeri negativi
	6	*	moltiplicazione
	6	/	divisione
	5	+	addizione
	5	-	sottrazione

TIPO	PRIORITA'	OPERATORE	COMMENTI
R	4	=	uguale
E			
L	4	<>	diverso
A			
Z	4	<	minore
I			
O	4		maggiore
N			
A	4	<=	minore o uguale
L			
I	4	>=	maggiore o uguale
L	3	NOT	complemento logico
O			(negazione)
G	2	AND	prodotto logico
I			
C	2	OR	somma logica
I			

Gli operatori relazionali e gli operatori logici servono a formare delle espressioni il cui valore e' una variabile, diciamo di tipo logico, cioe' un VERO o un FALSO; questa variabile assume valore aritmetico -1 se vale VERO e valore aritmetico 0 se vale FALSO.

Gli operatori logici servono a collegare due condizioni, le tabelle della verita' per essi sono:

AND

A	B	RISULTATO
vero	vero	vero
vero	falso	falso
falso	vero	falso
falso	falso	falso

OR

A	B	RISULTATO
vero	vero	vero
vero	falso	vero
falso	vero	vero
falso	falso	falso

NOT

A	RISULTATO
vero	falso
falso	vero

Nel confronto tra stringhe valgono per i caratteri alfabetici le normali regole dell'ordinamento alfabetico,

per gli altri caratteri l'ordinamento dipende dal valore dei corrispondenti codici ASCII.

I COMANDI

Si espongono qui i comandi del BASIC, che di norma vengono usati in modo immediato, per ognuno di essi si descrivera' l'effetto se usati all'interno di un programma. Tutti questi comandi lavorano quando si preme RETURN.

CONT serve per far proseguire un programma che si e' fermato. Esso non ha effetto se :

1) non esiste un programma;
2) si e' scritta una nuova linea di programma;
3) non si e' dato RUN;
4) c'e' stato un messaggio di errore.

Non ha senso usarlo in un programma.

LIST serve per listare sul video il programma che e'in memoria. Se usato da solo produce la lista di tutto il programma con eventuale scrolling. Lo scrolling puo' essere rallentato premendo CTRL o fermato premendo RUN STOP. Si puo' usare anche nei seguenti modi:

LIST n-	lista il programma dalla linea n in poi;
LIST n	lista solo la linea n;
LIST -n	lista dall'inizio fino alla linea n;
LIST n-m	lista dalla linea n alla linea m.

Se si usa in un programma, produce la lista e termina l'esecuzione del programma stesso.

LOAD serve per caricare in memoria un programma dal nastro o dal disco. Se scrivete solo LOAD il sistema cerca il primo programma che trova sul nastro e lo carica in memoria. Se scrivete LOAD "nome-programma" il sistema cerca sul nastro il programma avente quel nome e lo carica in memoria. Se scrivete LOAD "nome-programma",8 il sistema cerca sul disco il programma con quel nome e lo carica. Se non esiste sul dispositivo un programma con quel nome viene segnalato che non e' stato trovato. Il dispositivo cassetta corrisponde al numero di periferica #1, mentre il disco corrisponde al numero #8. In mancanza di virgola e numero dopo il nome del programma viene assunto 1 per il dispositivo.

Se il comando LOAD viene usato in un programma si ha il caricamento del nuovo programma al posto del vecchio e se questo non e' piu' lungo del precedente non vengono toccate le variabili che si trovano in memoria. Inoltre il nuovo programma va in esecuzione appena caricato. Per questa ragione si usa il LOAD da programma per ottenere il concatenamento tra programmi. E' quindi possibile segmentare

programmi lunghi per farli entrare in memoria.

NEW serve per cancellare il contenuto della memoria. E' buona norma usare questo comando prima di cominciare a scrivere un nuovo programma in memoria. Se usato da programma lascia la memoria pulita. Si deve fare attenzione e non usare NEW fuori tempo, si rischia di distruggere un lungo lavoro di scrittura di un programma in memoria.

RUN serve per mandare in esecuzione un programma che si trova in memoria. Prima dell'inizio del programma vengono azzerate le variabili. Se si scrive solo RUN il programma parte dalla linea con numero minore. SE si scrive RUN n, il programma parte dalla linea n, ma si ha lo stesso l'azzeramento delle variabili. Se si vuole far partire da una linea n un programma senza toccare le variabili, si deve usare il comando GOTO n. Non ha senso usare questo comando in un programma.

SAVE serve per memorizzare un programma che si trova in memoria o sulla cassetta o sul disco. Se si scrive solo SAVE il programma viene memorizzato sul nastro senza nome. Dovete fare attenzione e tenere la cassetta inattiva in modo che sia il sistema a chiedervi di avviarla, altrimenti rischiate di non aver posizionato bene un nastro e di cancellare altre registrazioni interessanti. Se scrivete SAVE seguito dal nome di un programma, viene registrato sul nastro il programma con il nome e questo e' sempre consigliabile. In tale modo poi le letture dei programmi si ottengono fornendo il nome del programma. Il comado puo' essere completato cosi':

SAVE "nome-programma",1,0	
SAVE "nome-programma"	questi due comandi sono e= quivalenti e memorizzano su cassetta;
SAVE "nome-programma",1,1	dopo aver memorizzato sulla cassetta scrive il carat= tere di FINE NASTRO;
SAVE "nome-programma",8	memorizza su disco .

Non ha senso dare questo comando da programma.

VERIFY serve per verificare quanto appena memorizzato comparandolo con il contenuto della memoria. Il comando va completato con il nome del programma e con il numero della periferica, a meno che non si tratti del nastro, nel qual caso si puo' tralasciare il numero e se si vuole anche il nome. Il comando VERIFY puo' essere usato con un nome non esistente per far scorrere il nastro e farlo posizionare dopo l'ultimo programma registrato. Non ha senso dare questo comando da programma.

LE ISTRUZIONI

Si espongono le istruzioni del BASIC. Tutte le istruzioni salvo,: INPUT, INPUT#, GET, GET# e DEFFN, possono essere usate sia in modo immediato che differito. Si ricorda che una linea di programma inizia con il numero di linea e puo' contenere una i struzione o piu' istruzioni e in quest'ultimo caso le istruzioni devono essere separate dai due punti.

CLOSE questo comando si scrive facendo seguire la parola chiave da un numero: CLOSE n. Dove n e' il numero assegnato logicamente al file nella OPEN iniziale. Per una corretta gestione dei files e' necessario chiudere tutti i file aperti prima di terminare un programma.

CLR si scrive senza parametri e serve per azzerare tutte le variabili di un programma, comprese le aree di lavoro come la stack area. Quando si fa RUN di un programma si ottiene automaticamente un CLR.

CMD con questo comando viene trasferito quanto normalmente esce sul video alla periferica appena aperta con lo stesso numero logico di file che segue il comando CMD. Esempio:

OPEN 1,4	apre la stampante che e' la periferica 4
CMD 1	trasferisce l'uscita al file di numero logico 1, appena aperto sulla stampante e quindi predispone la scrittura sulla stampante
LIST	manda sulla stampante la lista.

Per rimandare l'uscita al video si deve scrivere:

PRINT#1:CLOSE 1.

La struttura della frase e' uguale a quella della PRINT#.

DATA questa parola chiave deve essere seguita da una lista dei dati, separati da virgole, che si vogliono memorizzare nella zona apposita. In tale modo viene creato un magazzino di dati, dove i dati vengono memorizzati uno dopo l'altro con la stessa sequenza nella quale compaiono nei DATA. I dati numerici possono contenere il punto decimale. Le stringhe si possono scrivere senza apici delimitatori; e' necessario usare gli apici delimitatori se una stringa deve contenere i seguenti caratteri: spazio, virgola, due punti. Se nella lista dei dati compaiono due virgole (caratttere separatore dei dati) vicine il sistema assume che in mezzo sia o il carattere 0 o la stringa nulla. Non e' importante dove nel programma siano localizzate le frasi DATA, esse non sono frasi esecutive; e' consigliabile porle tutte insieme in fondo al programma. Esempio:

```
10DATA34,8.7,,PIPPO,6578,"SEGUONO: PRIMO, SECONDO","CASA   "
```

con questa frase DATA vengono memorizzate le seguenti costanti:

34	8.7	0	PIPPO
6578	SEGUONO: PRIMO, SECONDO		
CASA + 3 spazi			

Il sistema dopo aver preparato il blocco di dati posiziona un puntatore interno prima del primo dato; ogni volta che con la frase READ si preleva (legge) un dato dal blocco il puntatore viene spostato al dato seguente. Se si leggono piu' dati di quanti disponibili si ha un messaggio di errore. Il puntatore puo' essere riposizionato all'inizio del blocco dati con la frase RESTORE.

DEF FN serve a definire una funzione che puo' essere richiamata ed eseguita in un programma chiamandola per nome. Il nome della funzione deve seguire i caratteri FN ed essere seguito da una coppia di parentesi contenenti l'argomento della funzione, il simbolo = e poi la formula di calcolo; cosi':

```
10 DEF FNA(X)=13*(X-3)/(X+1)
```

Se nel programma si scrive:

```
50 PRINT FNA(9)
```

viene sostituito 9 ad X, effettuato il calcolo e stampato il valore risultante. Se invece si scrive:

```
80 Y = 57 + FNA(B)
```

viene sostituito ad X nella formula di calcolo il valore della variabile B ed Y viene posto al valore della funzione calcolata + 57.

DIM serve per definire le variabili con indice. Alla parola DIM segue la lista di definizione delle variabili, cosi':

```
DIM A1(4,5),B$(6,7,2)
```

Per calcolare il numero di elementi di ogni variabile si deve aggiungere 1 alle dimensioni esposte e moltiplicarle tra loro. Nell'esempio visto sopra A1 e' formato da (4+1)*(5+1)=30 elementi, mentre B$ e' formato da (6+1)*(7+1)*(2+1) elementi. In una frase DIM si possono definire tante variabili quante ne entrano nella linea. Non

esiste un limite teorico al numero delle dimensioni di una variabile con indice, ma tale limite e' dato dalla capacita' di memoria del calcolatore. Se il programma richiama una variabile con indice che non e' stata definita con una DIM, si ha un dimensionamento implicito a 10 (cioe' 11 elementi per ogni dimensione), se poi si pone la frase DIM piu' avanti nel programma si ha errore di ridimensionamento. Per questa ragione e' consigliabile scrivere tutte le frasi DIM all'inizio del programma.

Per quanto riguarda il dimensionamento delle stringhe si deve tener presente che al momento del dimensionamento il sistema crea solo le tabelle di riferimento alle stringhe con lo spazio per il puntatore al contenuto della stringa. Il contenuto della stringa occupa dinamicamente lo spazio riservato alle stringhe in memoria al momento dell'esecuzione del programma.

END fa terminare l'esecuzione del programma, senza messaggio di segnalazione. Il programma puo' continuare se si scrive CONT e se alla frase END seguono altre frasi nel programma.

FOR...TO...STEP serve per controllare l'esecuzione ripetitiva di un pezzo di programma. A questo comando e' legato il comando NEXT che serve per chiudere (delimitare) il ciclo da percorrere. Se si vuole creare un ciclo di attesa di alcuni secondi si puo' scrivere:

```
FOR K=1 TO N: NEXT K
```

e il tempo di attesa dipende dal valore di N.

Il formato della frase e':

```
FOR var.controllo = val.iniz. TO val.fin. STEP val.incr.
```

dove : var.controllo e' il nome della variabile che serve per controllare il ciclo (il contatore del ciclo);
- val.iniz. e' il valore iniziale per la variabile di controllo del ciclo;
- val.fin e' il valore finale per la variabile di controllo del ciclo;
- val.incr. e' il numero da sommare algebricamente alla variabile di controllo ad ogni ciclo.

Questi ultimi 3 dati possono anche essere variabili. Se l'incremento e' 1, si puo' tralasciare lo STEP.

Vengono eseguite in modo ciclico tutte le linee di programma comprese tra la frase FOR e la frase NEXT. Quando viene eseguita la frase FOR viene creata la variabile di controllo e le viene assegnato il valore iniziale, inoltre viene memorizzato il valore finale e l'incremento. Questo

significa che se questi due dati sono variabili, essi possono anche venir modificati nel ciclo. Nel ciclo invece non deve essere modificata la variabile di controllo. Poi il programma esegue le frasi che seguono FOR fino alla linea dove si trova NEXT. Quando viene incontrato il NEXT il sistema somma algebricamente lo STEP alla variabile di controllo e poi controlla se essa ha superato il valore finale. Se il valore finale non e' stato superato allora il programma torna ad eseguire le linee di programma che iniziano subito dopo il FOR. Se provate questo esempio vedete come funziona il FOR:

```
10 FOR K=1 TO 10
20 PRINT K;
30 NEXTK
35 PRINT
40 PRINT"VALORE DI K DOPO 10 GIRI: ";K
```

infatti vedrete sul video i numeri interi da 1 a 10 e poi a capo:

```
VALORE DI K DOPO 10 GIRI: 11
```

e questo sta a dimostrare che all'uscita dal ciclo K contiene un valore con il quale il ciclo non e' stato percorso.

Le frasi FOR possono essere concatenate tra loro, ma i cicli non possono intrecciarsi tra loro; segue uno schema di concatenamenti giusti e di concatenamento errato:

```
-----------        -----------          -----------
!                  !                    !
!------            !------              !------
!!                 !!                   !!
!!                 !-------             !!
!------            !                    -----------
!                  !----                 !
!                  !!                    !
-----------        !----                 ------
                   -----------

Concatenamenti corretti               Concat. errato
```

Questi schemi tradotti in frasi di programma corrispondono a:

```
10 FOR K=1 TO X      10 FOR K=1 TO Y      10 FOR K=1 TO Z
......               ......               ......
40 FOR J=1 TO N      40 FOR J=1 TO M      40 FOR J=1 TO R
......               ......               ......
80 NEXT J            70 NEXT J            90 NEXT K
......               ......               ......
90 NEXT K            80 FOR I= 1 TO       94 NEXT J
                     ......
                     90 NEXT 1
                     99 NEXT K
```

Si possono avere fino a 9 FOR concatenati, questa limitazione dipende dalle dimensioni dell' area stack.

GET in questa frase la parola chiave deve essere seguita dal nome di una variabile e serve per leggere un carattere dalla tastiera, senza dare il RETURN. Quando il programma esegue questa istruzione inizia immediatamente a leggere e quindi se non e' stato premuto alcun tasto legge una stringa nulla o uno zero a seconda del tipo di variabile che segue GET. Viene usata questa istruzione facendola seguire dall'analisi del carattere ricevuto, per creare delle attese nel programma ed operare delle scelte. Esempio:

```
10 GET A$
20 IF A$ = "" THEN 10
```

queste due istruzioni fanno proseguire il programma solo se viene premuto un qualunque tasto; infatti la stringa nulla non corrisponde ad alcun tasto.

GET# al carattere # deve seguire il numero logico di un file precedentemente aperto con OPEN e legge un carattere dal file.

GOSUB serve per saltare ad un sottoprogramma, la parola chiave deve essere seguita da un numero di linea. Il sistema ricorda il numero di linea della istruzione GOSUB e quando nel sottoprogramma incontra la frase RETURN torna alla linea dopo il GOSUB. Il sistema consente di avere al massimo 23 GOSUB attivi contemporaneamente.

GOTO serve per saltare ad una linea qualunque nel programma, e' l'istruzione di salto incondizionato. Si puo' anche scrivere GO TO.

IF.....THEN e' la frase che consente di operare delle scelte. Tra le parole chiave IF e THEN si pone una relazione condizionale; se essa e' vera il programma esegue quello che

viene chiesto dopo il THEN, se essa e' falsa il programma prosegue dalla linea seguente. Sono possibili 3 tipi di IF; li descriviamo facendo degli esempi:

```
. 100 IF A = B THEN 300
  110........
  ...
  ...
  300.......
```

se il valore di A e' uguale al valore di B, cioe' la condizione e' vera il pro= gramma prosegue dalla linea 300, se no prosegue da 110;

```
. 150 IF A<=B THEN PRINT "A<=B":GOTO 300
  160 PRINT "A>B"
  .....
  .....
  300....
```

se A<=B stampa A<=B e poi rimanendo sulla stessa li= nea esegue GOTO300, se no stampa A>B e prosegue;

```
. 100 IF A = B GOTO 100
```

e' analoga al primo esempio e si comporta allo stesso modo.

IF A THEN equivale a IF A <> 0 THEN.

Se dopo il THEN si vogliono eseguire piu' istruzioni esse devono essere separate dai due punti e vengono eseguite tutte se la condizione e' vera.

INPUT la parola chiave e' seguita dai nomi delle variabili nelle quali si vuole memorizzare quanto si legge dalla tastiera, i nomi delle variabili devono essere separati dalla virgola. Il programma si ferma ed evidenzia un punto interrogativo sul video. L'utente risponde con i dati e se ne scrive meno di quanti richiesti il programma si ferma ancora ed evidenzia 2 punti interrogativi per segnalare che e' ancora in attesa di dati. I dati che si scrivono devono essere separati dalla virgola e si deve terminare premendo RETURN. Se l'utente fornisce un numero di dati maggiore del numero delle variabili il sistema segnala EXTRA IGNORED ed il programma prosegue; i dati in piu' sono persi. Se la risposta e' errata per il tipo dei dati, cioe' a richiesta numerica si risponde con stringhe, il sistema segnala REDO FROM START e ripropone il punto interrogativo restando in attesa di INPUT.

Se si vuole rendere piu' comprensibile la richiesta di dati si puo' scrivere la frase ponendo dopo la parola INPUT una stringa tra apici recante una frase opportuna, scrivere un punto e virgola e poi la lista di variabili; cosi':

```
20 INPUT "SCRIVI 2 NUMERI";A,B
```

La frase tra apici non puo' superare i 20 caratteri.

La frase di INPUT puo' essere interrotta premendo contemporaneamente SHIFT destro e STOP RUN, oppure premendo contemporaneamente RUN STOP e RESTORE.

INPUT# lavora come la INPUT e legge dati dal file aperto avente lo stesso numero logico. Non ha senso in questo caso usare il formato con la frase tra apici.

LET questa parola chiave puo' essere usata nelle frasi di assegnazione e di calcolo, ma e' opzionale. Le frasi esempio che seguono hanno lo stesso significato:

```
10 LET A = 5      corrisponde a   10 A = 5
60 X = (X-3)*7    corrisponde a   60 LET X = (X-3)*7
```

NEXT questa frase serve per chiudere il ciclo di FOR. Alla parola chiave NEXT si puo' far seguire il nome della variabile di controllo del FOR o scrivere solo NEXT. Il sistema se manca la variabile chiude l'ultimo FOR aperto, se c'e' la variabile controlla che essa corrisponda a quella dell'ultimo FOR aperto. Si puo' anche scrivere NEXT L,K per chiudere due FOR di cui il primo aperto riguarda L ed il secondo K.

ON....GOTO.... questo comando consente le scelte multiple. Esso si scrive cosi':

```
ON K GOTO N1,N2,N3,....
```

e si comporta cosi:

. K, che puo' anche essere una espressione, viene considerata come numero intero;

. se K=1 va ad eseguire la linea numero N1, cioe' salta a N1, essendo N1 il primo numero nella lista;

. se K=2 va ad eseguire la linea N2, essendo N2 il secondo numero nella lista;

. e cosi' via tenendo conto del numero dei numeri di linea presenti;

. se K=0, K<0 o K maggiore del consentito il programma prosegue dalla linea seguente.

Si puo' usare anche: ON...GOSUB... e valgono le stesse regole viste per GOTO.

OPEN questa frase serve per permettere al VIC di collegarsi con le periferiche. Il formato completo della frase e':

```
OPEN LF,D,SA,FN
```

dove:

. LF e' il numero logico del file che si desidera aprire ed e' il numero da usare dopo # nelle frasi di lettura e scrittura, esso puo' andare da 1 a 255;

. D e' il numero che individua la periferica e puo' essere:

0 per il video;
1 per il registratore a nastro;
4 per la stampante;
8 per il disco;

. SA e' un indirizzo secondario che specifica a seconda della periferica usata il tipo di operazione, il numero del buffer, il canale usato. Per la cassetta:
SA=0 significa lettura, SA=1 significa scrittura, SA=2 significa scrittura con carattere finale di FINE FILE.

. FN e' il nome del file interessato all'operazione, non puo' essere omesso per il disco.

Esempi:

```
10 OPEN 1,0          apre il video come file numero 1;

15 OPEN 2,1,0,"DATI"   apre sulla cassetta il file
                       numero 2 di nome DATI per
                       leggere;

30 OPEN 3,4           apre sulla stampante il file 3;

40 OPEN 2,8,15       apre il file 2 sul canale 15 del
                     disco;

50 OPEN 2,8,2,"ANAGRAFE,S,W"
                     apre sul disco il file 2 sul buf=
                     fer 2 per scrivere (W) un file di
                     tipo sequenziale S.
```

POKE la parola chiave deve essere seguita da 2 numeri (o variabili o epressioni numeriche) separati da virgola. Il primo si riferisce all'indirizzo di una locazione di memoria e devè essere compreso tra 0 e 65535, mentre il secondo deve poter essere contenuto in un byte e quindi deve essere compreso tra 0 e 255. Il comando serve per scrivere nel byte di indirizzo "primo numero" il "secondo numero".

PRINT serve per fare apparire dati sul video. La parola chiave puo' essere seguita da:

. frasi tra apici;
. nomi di variabili;
. nomi di funzioni;
. simboli di punteggiatura come virgola e punto e virgola.

Gli elementi da stampare appaiono in sequenza sul video, andando a capo se la riga e' piena. La virgola fa saltare

per il prossimo elemento alla prossima zona di stampa se precedentemente sono stati stampati al massimo 10 caratteri, altrimenti viene saltata un'altra zona di stampa. La zona di stampa e' di 12 caratteri. Il punto e virgola fa stampare gli elementi vicini nel loro formato di stampa. Il formato di stampa per le stringhe non prevede aggiunte di caratteri, mentre per i numeri pone uno spazio o il segno prima e uno spazio dopo il numero.

Possono essere usate le funzioni TAB e SPC per posizionarsi dove si vuole.

Se la lista dei dati termina senza virgola o punto e virgola si ha una spaziatura verticale, se termina con uno dei due caratteri citati non si ha spaziatura verticale.

PRINT# il simbolo # deve essere seguito dal numero logico di un file precedentemento aperto. Per alcune apparecchiature periferiche non si possono usare le funzioni sopra viste.

READ la parola chiave e' seguita da una lista di variabili separate da virgole. Il programma va a prelevare dal blocco dati, partendo dalla posizione del puntatore interno i dati in sequenza e li assegna alle variabili della lista; si ha errore se si tenta di leggere un numero e si trova una stringa, o si tenta di leggere piu' dati di quelli disponibili.

REM alla parola chiave possono seguire commenti fino alla lunghezza massima di una linea di programma. Essa viene saltata durante l'esecuzione del programma.

RESTORE serve per riposizionare il puntatore interno all'inizio del blocco dei dati preparato dalle frasi DATA.

RETURN serve per chiudere logicamente un sottoprogramma e far ritornare alla linea dopo l'ultimo GOSUB eseguito.

STOP fa fermare il programma con il messaggio BREAK IN LINE ....; si puo' continuare il programma usando il comando CONT.

SYS deve essere seguito da un numero compreso tra 0 e 65535. Fa saltare il programma a questo indirizzo, dove deve essere stato memorizzato un programma in linguaggio macchina. Il programma in linguaggio macchina deve terminare con un opportuno comando per tornare al BASIC.

WAIT fa fermare il programma fino a quando il contenuto di una determinata locazione di memoria si e' modificato nel modo voluto. La frase si scrive cosi':

WAIT I,J,K analizza lo stato della locazione I;

esegue l'OR esclusivo con K;
esegue l'AND del risultato con J;
se il risultato e' diverso da 0
il programma prosegue, se no
ricomincia da capo.

LE FUNZIONI

Si riportano le funzioni del BASIC in ordine alfabetico divise nei 3 gruppi: numeriche, stringa, varie.

Numeriche:

ABS(X) fornisce il valore assoluto di x.

ATN(X) fornisce l'angolo in radianti la cui tangente e' X.

COS(X) fornisce il coseno dell'angolo X. X deve essere in radianti.

EXP(X) calcola il valore della costante e=2.71827183 elevata a X.

FNYY(X) calcola il valore della funzione YY, definita con DEF FNYY, con argomento X.

INT(X) ritorna il valore di X troncato dei decimali senza arrotondamento.

LOG(X) calcola il logaritmo naturale, cioe' in base e, di X. Per ottenere il log in base 10 basta dividere per LOG(10).

PEEK(X) fornisce il contenuto della locazione di memoria di indirizzo X. Deve essere X compreso tra 0 e 65535.

RND(X) questa funzione fornisce un numero pseudo-a-caso compreso tra 0 e 1. La sequenza dei numeri viene generata prendendo come origine un valore iniziale che dipende da X. Se X=0 la sequenza prende come origine il contatore dei fotogrammi dello schermo al momento. Se X<0 esso viene preso come origine e quindi ogni volta che si fa girare il programma si ottiene la stessa sequenza di numeri. Se X>0 il valore di X non influisce sull'origine della sequenza, che dipende dallo stato del calcolatore.

SGN(X) fornisce 0 per X=0, 1 per X>0 e -1 per X<0.

SIN(X) calcola il seno dell'angolo X, che deve essere espresso in radianti.

SQR(X) calcola la radice quadrata di X, che deve essere positivo, altrimenti si ha errore.

TAN(X) calcola la tangente dell'angolo X che deve essere in radianti.

USR(X) fa saltare il programma ad una sequenza di istruzioni in linguaggio macchina il cui indirizzo iniziale si deve trovare nei bytes 1 e 2. Il valore di X viene passato al programma in linguaggio macchina e questo ritorna un valore al programma Basic.

Stringa:

ASC(X$) ritorna il codice ASCII del primo carattere della stringa X$.

CHR$(X) e' la funzione opposta di ASC, fornisce il carattere il cui codice ASCII e' X.

LEFT$(X$,X) ritorna una stringa formata dagli X caratteri di sinistra della stringa X$.

LEN(X$) fornisce un numero che e' uguale al numero di caratteri della stringa.

MID$(X$,S,X) ritorna una stringa formata da X caratteri della stringa X$ partendo dalla posizione S.

RIGHT$(X$,X) ritorna gli X caratteri piu' a destra della stringa X$.

STR$(X) ritorna il numero X trasformato in stringa, preceduto da uno spazio o dal segno meno.

VAL(X$) e' l'inverso della funzione precedente. La stringa X$ viene trasformata in un numero. Se la stringa contiene caratteri non numerici la conversione avviene per i caratteri numerici validi che precedono i non validi senza segnalazione di errore.

Varie:

FRE(X) non ha importanza cosa contiene X, la funzione fornisce un numero uguale al numero di bytes liberi nella memoria del VIC.

POS(X) ritorna un numero compreso tra 0 e 21 che rappresenta la colonna sulla quale si trova il cursore dello

schermo; X puo' essere qualunque.

SPC(X) fa saltare X spazi senza cancellare eventuali caratteri presenti.

TAB(X) X rappresenta la colonna dove si vuole sia stampato il prossimo carattere. Se X supera 21 si passa alla riga successiva.

APPENDICE F

LE POKE DI USO COMUNE E ALTRE COSE UTILI

Per comodita' riportiamo un elenco dei comandi POKE piu' usati normalmente:

POKE 36869,240	fa passare al set di caratteri grafici, quello attivo al mo= mento dell'accensione
POKE 36869,242	fa passare al set di caratteri maiuscolo/minuscolo
POKE 36879,X	modifica in base a X la combi nazione dei colori dello sfon= do, del bordo e del testo
POKE 36879,27	attiva la combinazione di colo= ri che si ha all'accensione
POKE 36878,X	predispone il volume per il so= noro, 0<=X<=15
POKE 36877,X	serve per la tonalita' del ru= more bianco, 128<=X<=255
POKE,36876,X	serve per la tonalita' alta, 128<=X<=255
POKE 36875,X	serve per la tonalita' media, 128<=X<=255
POKE 36874,X	serve per la tonalita' bassa, 128<=X<=255

Riportiamo un elenco delle CHR$ piu' usate con il comando PRINT:

CHR$(5)	predispone il testo in bianco
CHR$(13)	sostituisce il RETURN
CHR$(14)	fa passare al set maiuscolo/minusco= lo
CHR$(17)	fa abbassare il cursore di una riga

CHR$(18)	attiva il RVS ON
CHR$(19)	sposta il cursore nell' angolo in alto a sinistra
CHR$(20)	corrisponde al tasto DEL
CHR$(28)	predispone il testo in rosso
CHR$(29)	sposta il cursore a destra di una co lonna
CHR$(30)	predispone il testo in verde
CHR$(31)	predispone il testo in blu
CHR$(32)	produce uno spazio
CHR$(141)	corrisponde a SHIFT e RETURN
CHR$(142)	fa passare al set grafico
CHR$(144)	predispone il testo in nero
CHR$(145)	fa alzare il cursore di una riga
CHR$(146)	disattiva il RVS ON cioe' corrispon= de a RVS OFF
CHR$(147)	pulisce lo schermo e manda il curso= re in alto a sinistra
CHR$(148)	corrisponde al tasto INST
CHR$(156)	predispone il colore in viola
CHR$(157)	fa spostare il cursore a sinistra di una posizione
CHR$(158)	predispone il testo in giallo
CHR$(159)	predispone il testo in azzurro
CHR$(160)	produce uno spazio + shift

Ricordiamo inoltre che:

SPC(X)	fa spaziare di X posizioni senza cancellare

TAB(X) fa andare alla colonna X, la prima a sinistra corrisponde a zero

Per interrompere un programma che chiede INPUT si devo= no premere contemporaneamente i tasti RUN STOP e RESTO= RE, oppure rispondere solo con RETURN.

RUN STOP e SHIFT fa caricare il primo programma che si trova sulla cassetta.

Per ottenere in stampa numeri interi incolonnati a destra si deve procedere cosi':

- se i numeri sono interi trasformarli in stringa con la funzione STR$(K), la quale pone davanti al nume= ro uno spazio o il segno meno;
- preparare una stringa di spazi shiftati, cioe' dopo gli apici premere un certo numero di spazio e SHIFT insieme e chiudere gli apici;
- concatenare la stringa di spazi con la stringa del numero e prendere con la funzione RIGHT$ il numero di caratteri desiderato.

Esempio:
```
A%=546
B%= -3458
A$=STR$(A%)
B$=STR$(B%)
S$=" 6 spazi + SHIFT"
PRINT RIGHT$(S$+A$,9)
PRINT RIGHT$(S$+B$,9)
```
vedrete apparire i numeri incolonnati a destra.

Per ottenere lo stesso risultato con i numeri decimali dovete moltiplicarli per una opportuna potenza di 10 (1000 se volete 3 decimali, 100 se ne volete 2), poi renderli interi con la funzione INT. A questo punto i numeri possono essere trasformati in stringa mediante la STR$, la stringa puo' essere manipolata separando la parte decimale dalla intera, si puo' inserire il punto o la virgola decimale come stringa e operare co= me sopra per incolonnarli.

Ricordate che in fase di INPUT gli spazi a sinistra del dato vengono cancellati, se volete conservarli dovete usare spazio + SHIFT.

Per avere stringhe tutte della stessa lunghezza si deve creare una stringa di spazi o di spazi + SHIFT e operare dei concatenamenti, poi con le funzioni di stringa si prendono le parti che interessano.

I COMANDI ABBREVIATI

Abbiamo gia' sperimentato che invece della parola PRINT si puo' usare il punto interrogativo; questo e' un modo abbreviato di scrivere il comando. Esistono possibilita' analoghe anche per altri comandi BASIC e sono riportate nella tabella che segue. Per abbreviare i comandi si deve scrivere o solo la prima lettera seguita (o le prime due lettere seguite) dalla prossima lettera premuta insieme al tasto SHIFT. Nel programma viene conservato il codice corrispondente alla parola chiave completa, come spiegato nella Appendice J. Se si chiede la lista del programma con il comando LIST le parole chiave appaiono complete. In fase di scrittura quando si preme una lettera insieme a SHIFT appare il corrispondente carattere grafico.

COMANDO	ABBREVIAZIONE	COMANDO	ABBREVIAZIONE
AND	A + SHIFT e N	SAVE	S + SHIFT e A
NOT	N + SHIFT e O	STEP	ST + SHIFT e E
CLOSE	CL + SHIFT e O	STOP	S + SHIFT e T
CLR	C + SHIFT e L	SYS	S + SHIFT e Y
CMD	C + SHIFT e M	THEN	T + SHIFT e H
CONT	C + SHIFT e O	VERIFY	V + SHIFT e E
DATA	D + SHIFT e A	WAIT	W + SHIFT e A
DEF	D + SHIFT e E	ABS	A + SHIFT e B
DIM	D + SHIFT e I	ASC	A + SHIFT e S
END	E + SHIFT e N	ATN	A + SHIFT e T
FOR	F + SHIFT e O	CHR$	C + SHIFT e H
GET	G + SHIFT e E	EXP	E + SHIFT e X
GOSUB	GO + SHIFT e S	FRE	F + SHIFT e R
GOTO	G + SHIFT e O	LEFT$	LE + SHIFT e F
INPUT#	I + SHIFT e N	MID$	M + SHIFT e I
LET	L + SHIFT e E	PEEK	P + SHIFT e E
LIST	L + SHIFT e I	RIGHT$	R + SHIFT e R
LOAD	L + SHIFT e O	RND	R + SHIFT e N
NEXT	N + SHIFT e E	SGN	S + SHIFT e G
OPEN	O + SHIFT e P	SIN	S + SHIFT e I
POKE	P + SHIFT e O	SPC(	S + SHIFT e P
PRINT	?	SQR	S + SHIFT e Q
PRINT#	P + SHIFT e R	STR$	ST + SHIFT e R
READ	R + SHIFT e E	TAB(	T + SHIFT e A
RESTORE	RE + SHIFT e S	USR	U + SHIFT e S
RETURN	RE + SHIFT e T	VAL	V + SHIFT e A
RUN	R + SHIFT e U		

APPENDICE H

COME CALCOLARE ALCUNE FUNZIONI MATEMATICHE

Le funzioni che seguono non sono fornite intrinsecamente dal BASIC, ma possono essere calcolate con le seguenti formule:

Funzione	Formula di calcolo
SECANTE	SEC(X)=1/COS(X)
COSECANTE	CSC(X)=1/SIN(X)
COTANGENTE	COT(X)=1/TAN(X)
ARCOSENO	ARCSIN(X)=ATN(X/SQR(-X*X+1))
ARCOCOSENO	ARCCOS(X)=-ATN(X/SQR(-X*X+1)) +<PI>/2
ARCOSECANTE	ARCSEC(X)=ATN(X/SQR(X*X-1))
ARCOCOSECANTE	ARCCSC(X)=ATN(X/SQR(X*X-1)) +(SGN(X)-1)*<PI>/2
ARCOCOTANGENTE	ARCOT(X)=ATN(X)+<PI>/2
SENOIPERBOLICO	SINH(X)=(EXP(X)-EXP(-X))/2
COSENOIPERBOLICO	COSH(X)=(EXP(X)+EXP(-X))/2
TANGENTEIPERBOLICA	TANH(X)=EXP(-X)/(EXP(X)+EXP(-X)) *2+1
ARCOSENOIPERBOLICO	ARCSINH(X)=LOG(X+SQR(X*X+1))
ARCOCOSENOIPERBOLICO	ARCCOSH(X)=LOG(X+SQR(X*X-1))
ARCOTANGENTEIPERBOLICA	ARCTANH(X)=LOG((1+X)/(1-X))/2
ARCOSECANTEIPERBOLICA	ARCSECH(X)=LOG((SQR(-X*X+1)+1)/X)
ARCOCOSECANTEIPERBOLICA	ARCCSCH(X)=LOG((SGN(X)*SQR(X* X+1)/X)
ARCOCOTANGENTEIPERBOLICA	ARCCOTH(X)=LOG((X+1)/(X-1))/2

APPENDICE I

COLORI SFONDO/BORDO E TESTO

Al momento dell'accensione del VIC il colore dello sfondo e' bianco, il colore del bordo e' azzurro ed il cursore e' blu.

Se volete modificare la combinazione dei colori dovete scrivere ed eseguire o in modo immediato o in una linea di programma:

```
POKE 36879,X
```

dove X deve essere un numero compreso tra 0 e 255.

Per facilitarvi nella scelta dei colori potete servirvi della tabella che segue, dove sono riportati i numeri corrispondenti alle combinazioni dei colori sfondo/bordo, da usare come X nel comando POKE. Nelle colonne della tabella sono riportati gli 8 possibili colori del bordo e nelle righe i 16 possibili colori dello sfondo.

	BORDO							
SFONDO	BLK	WHT	RED	CYAN	PUR	GRN	BLU	YEL
BLACK	8	9	10	11	12	13	14	15
WHITE	24	25	26	27	28	29	30	31
RED	40	41	42	43	44	45	46	47
CYAN	56	57	58	59	60	61	62	63
PURPLE	72	73	74	75	76	77	78	79
GREEN	88	89	90	91	92	93	94	95
BLUE	104	105	106	107	108	109	110	111
YELLOW	120	121	122	123	124	125	126	127
ORANGE	136	137	138	139	140	141	142	143
LT. ORANGE	152	153	154	155	156	157	158	159
PINK	168	169	170	171	172	173	174	175
LT. CYAN	184	185	186	187	188	189	190	191
LT. PURPLE	200	201	202	203	204	205	206	207
LT. GREEN	216	217	218	219	220	221	222	223
LT. BLUE	232	233	234	235	236	237	238	239
LT. YELLOW	248	249	250	251	252	253	254	255

Come potete vedere i numeri colore vanno di 8 in 8 in

sequenza, i numeri che mancano producono la stessa combinazione di colori sfondo/bordo, ma con il testo in campo inverso, usando la formula:

X = X - 8

Dopo aver determinato i colori dello sfondo e del bordo, potete modificare il colore del testo in uno dei modi che seguono:

. premendo contemporaneamente il tasto CTRL ed uno dei tasti colore;

. eseguendo un comando PRINT seguito da CHR$(Y), dove Y e' il codice colore prelevato dalla tabella dei codici ASCII della Appendice A;

. eseguendo un comando PRINT seguito tra apici dal carattere colore ottenuto premendo contemporanramente il tasto CTRL e il tasto colore;

. eseguendo un comando POKE che scrive il codice colore in modo diretto nei bytes della mappa colori dello schermo, ricordando che in tale caso il codice colore va da 0 a 7, cioe' equivale al numero scritto sul tasto colore -1.

Abbiamo visto che togliendo 8 al numero colore si ottiene il campo inverso. Vediamo il contenuto del byte 36879 in binario:

Posizione dei bits

Tenendo conto che il semibyte di sinistra (bits 7, 6, 5 e 4) ha peso 16 e che il bit 3 ha peso 8, si capisce il perche' della formula per calcolare il contenuto del byte 36879:

C = S * 16 + 1 * 8 + B per campo diretto e

C = S * 16 + 0 * 8 + B per campo inverso,

dove: C = numero colore per byte 36879;
S = numero colore per sfondo;
B = numero colore per bordo.

A questo punto, dopo aver fatto parecchie prove, vi chiederete cosa succede quando, in uno dei modi visti sopra si modifica il colore del testo.

Nelle 128 combinazioni di colore sfondo/bordo dirette il cursore e' blu, come nella situazione di accensione iniziale a meno che non se ne cambi il colore in uno dei modi sopra descritti per il colore del testo. Se lo sfondo e' blu (o dello stesso colore del cursore in quel momento) non si vede il testo.

Nelle 128 combinazioni in campo inverso lo sfondo e' bianco, mentre il colore dello sfondo viene assunto dal testo che appare in campo inverso su uno sfondo (localizzato al carattere) determinato dal colore iniziale del cursore a meno che non si sia predisposto un cambiamento di colore del testo, come visto sopra.

In questo modo si arriva ad avere presenti sullo schermo diversi colori, ma attivi contemporaneamente solo 3 colori nel modo diretto e 4 nel modo inverso.

In realta' il VIC puo' funzionare anche in un altro modo, cioe' con piu' colori, riducendo pero' la risoluzione dello schermo. Di questo ci occuperemo nel prossimo manuale.

APPENDICE J

EQUIVALENTI DECIMALI DELLE PAROLE CHIAVE E DEI SIMBOLI DEL BASIC

Nella tabella che segue sono riportati gli equivalenti decimali delle parole chiave e dei simboli del BASIC. I codici numerici riportati vengono memorizzati in 1 byte e, come potete osservare hanno valori compresi tra 128 e 203. Tali valori come caratteri ASCII (riportati nella Appendice A) hanno un altro significato; l'interpretazione del codice numerico dipende dal contesto nel quale si trova.

Facciamo un esempio:

. il codice 161 corrisponde alla parola chiave GET e a un carattere grafico;

. se l'interprete BASIC trova 161 dopo il numero di linea o dopo un due punti, questo codice viene interpretato come GET;

. se invece 161 viene trovato dopo il codice del doppio apice viene interpretato come carattere grafico.

Come potete osservare nel codice di TAB ed SPC e' contenuta anche la prima parentesi.

Nella pagina seguente e' riportata la tabella dei codici.

TABELLA DEI CODICI

128	END	166	SPC(
129	FOR	167	THEN
130	NEXT	168	NOT
131	DATA	169	STEP
132	INPUT#	170	+
133	INPUT	171	-
134	DIM	172	*
135	READ	173	/
136	LET	174	
137	GOTO	175	AND
138	RUN	176	OR
139	IF	177	>
140	RESTORE	178	=
141	GOSUB	179	<
142	RETURN	180	SGN
143	REM	181	INT
144	STOP	182	ABS
145	ON	183	USR
146	WAIT	184	FRE
147	LOAD	185	POS
148	SAVE	186	SQR
149	VERIFY	187	RND
150	DEF	188	LOG
151	POKE	189	EXP
152	PRINT#	190	COS
153	PRINT	191	SIN
154	CONT	192	TAN
155	LIST	193	ATN
156	CLR	194	PEEK
157	CMD	195	LEN
158	SYS	196	STR$
159	OPEN	197	VAL
160	CLOSE	198	ASC
161	GET	199	CHR$
162	NEW	200	LEFT$
163	TAB(	201	RIGHT$
164	TO	202	MID$
165	FN	203	GO

APPENDICE K

EFFETTI SONORI

Si riportano i registri utilizzati per il sonoro ed i valori approssimativi da usare per ottenere le note musicali.

Elenco dei comandi POKE per modificare i registri del sonoro:

POKE 36878,X	predispone il volume del suono, X puo' variare da 0 a 15; dato che il semibyte di ordine piu' elevato puo' essere impegnato per il multicolore, fare attenzione;
POKE 36874,X	predispone la prima tonalita' e X puo' variare da 128 a 255;
POKE 36875,X	predispone la seconda tonalita' e X puo' variare da 128 a 255;
POKE 36876,X	predispone la terza tonalita' e X puo' variare da 128 a 255;
POKE 36877,X	predispone il generatore di rumore bianco e X puo' variare da 128 a 255.

Elenco dei valori approssimativi da usare per le note musicali:

NOTA	VALORE	NOTA	VALORE
DO	135	SOL	215
DO#	143	SOL#	217
RE	147	LA	219
RE#	151	LA#	221
MI	159	SI	223
FA	163	DO	225
FA#	167	DO#	227
SOL	175	RE	228
SOL#	179	RE#	229
LA	183	MI	231
LA#	187	FA	232
SI	191	FA#	233
DO	195	SOL	235
DO#	199	SOL#	236
RE	201	LA	237
RE#	203	LA#	238
MI	207	SI	239
FA	209	DO	240
FA#	212	DO#	241

APPENDICE L

CONVERSIONI NUMERICHE

Per convertire un numero decimale in binario si puo' procedere in uno dei 2 modi seguenti.

MODO 1:

. tracciare una linea orizzontale ed un certo numero di linee verticali;
. scrivere il numero da convertire sopra la linea orizzontale a sinistra della prima verticale;
. procedere a successive divisioni per due scrivendo il quoziente nella prossima posizione sopra la riga ed il resto sotto il dividendo;
. fermarsi quando il quoziente e' zero;
. i resti letti da destra verso sinistra danno il numero binario.

Esempio: trasformazione di 3715 in binario.

```
3715!1857! 928! 464! 232! 116! 58! 29! 14! 7! 3! 1!0
----!----!----!----!----!----!---!---!---!--!--!--!--
   1!   1!   0!   0!   0!   0!  0!  1!  0! 1! 1! 1!
```

Si ottiene 111010000011

MODO 2:

. confrontare il numero da convertire con le potenze di 2 e sottrargli la piu' alta potenza di 2 minore del numero;
. porre un 1 nella posizione che nel numero binario compete a quella potenza;
. sottrarre la potenza dal numero;
. procedere cosi' fino ad ottenere zero;
. completare le posizioni rimaste vuote del numero binario con degli zeri.

Esempio: trasformazione di 3715 in binario.

```
 3715-        bit   11 10  9  8  7  6  5  4  3  2  1  0
 2048=              ------------------------------------
------
 1667                1

 1667-
 1024=
------
  643                   1

  643-
  512=
------
  131                      1

  131-
  128=
------
    3                            1

    3-
    2=
-------
    1                                              1  1

e completando        1  1  1  0  1  0  0  0  0  0  1  1
```

POTENZE DEL 2

Esponente	Valore	Esponente	Valore
0	1	16	65536
1	2	17	131072
2	4	18	262144
3	8	19	524288
4	16	20	1048576
5	32	21	2097152
6	64	22	4194304
7	128	23	8388608
8	256	24	16777216
9	512	25	33554432
10	1024	26	67108864
11	2048	27	134217728
12	4096	28	268435456
13	8192	29	536870912
14	16384	30	1073741824
15	32768	31	2147483648

Per convertire da binario a decimale basta moltiplicare

bit 1 per le corrispondenti potenze di 2 e sommare.

Per convertire da decimale a esadecimale si puo' usare il metodo visto sopra delle divisioni successive, prendendo i resti invertiti.

Un numero esadecimale si trasforma molto rapidamente in binario tenendo conto che:

Esadecimale	Binario	Esadecimale	Binario
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

da ogni cifra esadecimale nascono 4 bit.

La trasformazione inversa si ottiene raggruppando i bit a 4 a 4 e sostituendo la corrispondente cifra esadecimale.

Se si leggono con la funzione PEEK i valori dei byte e si deve calcolare il valore di 2 byte associati si deve procedere (in decimale) cosi':

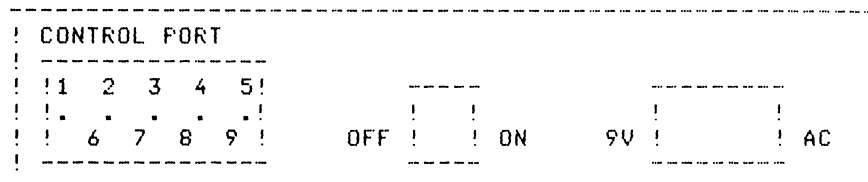
HI * 256 + LO   dove HI significa byte piu' significativo e LO significa byte meno significativo.

I numeri decimali sono conservati in memoria nella forma floating point normalizzata, cioe' vengono conservate le cifre significative del numero a partire dalla prima e perdendo eventualmente quelle meno significative, e viene conservato l'esponente da dare a 10 per ottenere il valore del numero. L'esponente prende il nome di caratteristica del numero e le cifre prendono il nome di mantissa. Il calcolatore naturalmente conserva i numeri in binario e usa 5 byte per la mantissa e 1 byte per la caratteristica. Il segno del numero viene conservato nella mantissa con la stessa convenzione usata per i numeri interi, mentre per il segno della caratteristica viene usato il seguente accorgimento:

. il valore del byte viene diminuito di 128;
. gli esponenti positivi sono da 1 a 128 (dopo aver sottratto 128);
. gli esponenti negativi sono da -127 a 0 (sempre dopo la sottrazione).

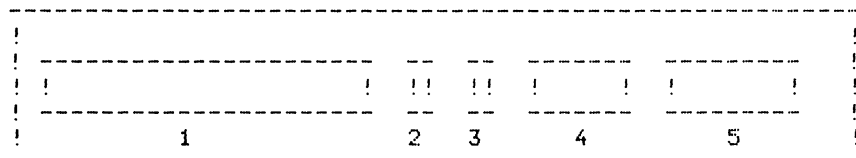
SCHEMI COLLEGAMENTI

Connessioni laterali del VIC:

```
-------------------------------------------------------------------------
! CONTROL PORT                                                          !
! --------------                                                        !
! !1  2  3  4  5!          -----            ---------                   !
! !.  .  .  .  .!         !    !            !        !                  !
! !  6  7  8  9 !     OFF !    ! ON     9V  !        ! AC               !
! ---------------          -----            ---------                   !
-------------------------------------------------------------------------
```

I/O Giochi

PIN#	TIPO	NOTE
1	JOY0	
2	JOY1	
3	JOY2	
4	JOY3	
5	POT Y	
6	LIGHT PEN	
7	+5V	MAX.100mA
8	MASSA	
9	POT X	

Connessioni posteriori del VIC:

```
-----------------------------------------------------------------------
!                                                                     !
! ---------------------   --  --  -------   --------                  !
! !                   !   !!  !!  !     !   !      !                  !
! ---------------------   --  --  -------   --------                  !
!           1              2   3     4          5                     !
-----------------------------------------------------------------------
```

1) ESPANSIONE MEMORIA

```
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
-------------------------------------------------------------
!. . . . . . . . . .  .  .  .  .  .  .  .  .  .  .  .  .   !
!                                                          !
!                                                          !
!. . . . . . . . . .  .  .  .  .  .  .  .  .  .  .  .  .   !
-------------------------------------------------------------
 A B C D E F H J K L  M  N  P  R  S  T  U  V  W  X  Y  Z
```

PIN#	TIPO	PIN#	TIPO	PIN#	TIPO	PIN#	TIPO
1	MASSA	12	BLK3	A	MASSA	N	CA10
2	CD0	13	BLK5	B	CA0	P	CA11
3	CD1	14	RAM1	C	CA1	R	CA12
4	CD2	15	RAM2	D	CA2	S	CA13
5	CD3	16	RAM3	E	CA3	T	I/02
6	CD4	17	VR/W	F	CA4	U	I/03
7	CD5	18	CR/W	H	CA5	V	S/02
8	CD6	19	IRQ	J	CA6	W	NMI
9	CD7	20	NC	K	CA7	X	RESET
10	BLK1	21	+5v	L	CA8	Y	NC
11	BLK2	22	MASSA	M	CA9	Z	MASSA

2) AUDIO/VIDEO

```
------------------
!                !
!   3        1   !
!   .        .   !
!    5     4     !
!    .     .     !
!       2        !
!       .        !
------------------
```

PIN#	TIPO	NOTE
1	+6V	MAX.10mA
2	MASSA	
3	AUDIO	
4	VIDEO LOW	
5	VIDEO HIGH	

3) I/O SERIALE

```
------------------
!                !
!    5      1    !
!    .      .    !
!       6        !
! 4     .     2  !
! .           .  !
!       3        !
!       .        !
------------------
```

PIN#	TIPO
1	SRQ IN SERIALE
2	MASSA
3	ATN IN/OUT SERIALE
4	CLK IN/OUT SERIALE
5	DATA IN/OUT SERIALE
6	NC

4) REGISTRATORE A NASTRO

```
  1  2  3  4  5  6
  .  .  .  .  .  .
-----------------------
!                     !
!                     !
-----------------------
  .  .  .  .  .  .
  A  B  C  D  E  F
```

PIN#	TIPO
A-1	MASSA
B-2	+5V
C-3	MOTORE REGISTRATORE
D-4	LETTURA
E-5	SCRITTURA
F-6	INTERRUTORE

5) PORTE UTENTE (Ingresso/Uscita)

```
  1   2   3   4   5   6   7   8   9  10  11  12
  .   .   .   .   .   .   .   .   .   .   .   .
-----------------------------------------------------
!                                                   !
-----------------------------------------------------
  .   .   .   .   .   .   .   .   .   .   .   .
  A   B   C   D   E   F   H   J   K   L   M   N
```

PIN#	TIPO	NOTE	PIN#	TIPO
1	MASSA		A	MASSA
2	+5V	MAX.100mA	B	CB1
3	RESET		C	PB0
4	JOY0		D	PB1
5	JOY1		E	PB2
6	JOY2		F	PB3
7	LIGHT PEN		H	PB4
8	INTERR.REG.		J	PB5
9	ATN IN SERIALE		K	PB6
10	+9V	MAX.100mA	L	PB7
11	MASSA		M	CB2
12	MASSA		N	MASSA

APPENDICE N

LE ESTENSIONI DEL VIC

Sono previste le seguenti estensioni:

. Cartucce programmate:

. Super espansione - aggiunge 3K di memoria RAM (raggiungendo cosi' 8k)
- alta risoluzione grafica e comandi di plottaggio
- funzioni chiave pre-assegna= te.

. Programmi utilita'- nuove funzioni Basic
- Monitor di linguaggio mac= china
- funzioni chiave pre-assegna= te
- funzioni assegnabili dall'u= tente.

. Interfaccia principale di collegamento:

si inserisce nella porta di espansione. Si possono collegare fino a 6 dispositivi: cartucce, espansio= ne di memoria, interfaccia IEEE-488 per periferiche.

. Unita' a singolo drive per floppy disk:

la capacita' del disco arriva a 170 mila caratteri; si inserisce al BUS seriale.

. Cartuccia di interfaccia IEEE-488:

consente di collegare le periferiche Commodore e strumenti scientifici.

. Stampante seriale:

permette di collegare una stampante seriale.

ESEMPI DI PROGRAMMI

PROGRAMMA VOLO UCCELLINO

In questo programma i caratteri di controllo per i comandi vengono generati mediante la funzione CHR$. Nella prima parte, dalla linea 10 alla linea 94 si preparano le stringhe con i caratteri di controllo e una stringa di 15 spazi, necessaria quando si usano le funzioni di stringa.
La linea 100 predispone il colore dello sfondo e del bordo.
Le linee da 105 a 155 disegnano le nuvole nel cielo.
Le linee da 160 a 185 preparano le stringhe che danno il disegno dell'uccellino.
Le linee da 190 a 250 fanno volare l'uccellino dall'angolo in alto a sinistra all'angolo in basso a destra.
Le linee da 260 a 280 modificano la direzione del volo.
Il programma non si ferma mai, si puo' interromperlo con STOP RUN.

Segue la lista del programma:

```
 10 REM PREPARAZIONE STRINGHE
 20 REM GRAFICHE E DI CONTROLLO
 25 S0$=CHR$(19)
 30 S1$=CHR$(147)
 35 S2$=CHR$(144)
 40 S3$=CHR$(5)
 45 S4$=CHR$(18)
 50 S5$=CHR$(146)
 55 S6$=CHR$(17)
 60 S7$=CHR4(113)
 65 SA$=CHR$(105)
 70 SB$=CHR$(117)
 75 SC$=CHR$(106)
 80 SD$=CHR$(107)
 85 S8$=CHR$(127)
 90 S9$=CHR$(169)
 91 CS$=CHR$(157)
 92 CD$=CHR$(29)
 93 CU$=CHR$(145)
 94 SP$="               ":REM 15 SPAZI
 95 REM COLORE SFONDO E SCHERMO AZZURRO
100 POKE 36879, 59
105 REM DISEGNO DELLE NUVOLE
```

```
110 PRINT S1$;S3$
115 PRINT S6$+S6$+S6$+S6$;SP$;S4$;S9$;LEFT$(SP$,6)
120 PRINT S0$;S6$+S6$+S6$+S6$+S6$;LEFT$(SP$,14);
121 PRINT S4$;S9$;LEFT$(SP$,7)
125 PRINT S0$;S6$+S6$+S6$+S6$+S6$+S6$;LEFT$(SP$,14);
126 PRINT S4$;LEFT$(SP$,8)
130 PRINT S0$;S6$+S6$+S6$+S6$+S6$+S6$+S6$;
131 PRINT LEFT$(SP$,14);S4$;LEFT$(SP$,7)
135 PRINT S0$;S6$+S6$+S6$+S6$+S6$+S6$+S6$+S6$;
136 PRINT SP$;S8$;S4$;LEFT$(SP$,6)
140 PRINT S6$+S6$;S4$;LEFT$(SP$,5);S8$
145 PRINT S4$;LEFT$(SP$,7)
150 PRINT S4$;LEFT$(SP$,5);S5$;S9$
155 PRINT S4$;LEFT$(SP$,3);S5$;S9$
160 REM MOVIMENTO UCCELLINO
165 B$=CS$+CS$+CS$
170 PRINT S0$;S2$
175 U$(1)=S6$+CD$+SC$+S7$+SD$
180 U$(2)=SB$+S7$+SA$
185 U$(3)=SPC(3)
190 REM VOLO UCCELLINO
200 FOR M=1 TO 18
210 FOR J=1 TO 4
220 IF U$(J)= "" THEN 250
230 PRINT U$(J);B$;
240 FOR T=1 TO 200:NEXTT
250 NEXTJ:NEXTM
260 REM CAMBIO DIREZIONE VOLO
270 IF V=1 THEN V=0:GOTO 175
280 U$(1)=CS$+CU$+RIGHT$(U$(1),3):V=1:GOTO 190
```

PROGRAMMA SCONTRO AUTOMOBILINE

In questo programma i caratteri di controllo sono esposti racchiudendoli tra i simboli di minore e maggiore (<....>). Il tasto chiamato COM e' quello situato in basso a sinistra sulla tastiera.

Segue la lista del programma:

```
 1 CD$="<CRSR verticale>"
 3 CL$="<SHIFT e CRSR laterale>"
 5 PRINT "<SHIFT e CLR HOME>"
10 G$="<CLR HOME>"+CD$+CD$+CD$+CD$+CD$+CD$+CD$+CD$+
       CD$+CD$
20 A$=CL$+" <RVS ON><CTRL e 3><SHIF e lira ste
```

```
            rlina> <RVS OFF><COM e I>"
 25 C$=CL$+"  <SHIFT e Q> <SHIFT e Q>"
 30 B$="<CTRL e 6><COM e I><RVS ON> <COM e *><RVS OFF> "
 35 D$="<SHIFT e Q> <SHIFT e Q> "
 40 FOR K=0 TO 228
 50 PRINT G$;SPC(239-K);B$;SPC(18);D$;
 60 PRINT "<CLR HOME>";SPC(K);A$;SPC(18);C$;
 70 FOR I=0 TO 10:NEXTI
 80 NEXTK
 90 A=28:M=0
100 PRINT CHR$(A);
105 PRINT G$;SPC(10);"<SHIFT e M> <SHIFT e N>"
110 PRINT SPC(10);"   "
115 PRINT SPC(10);"<SHIFT e N> <SHIFT e M>"
118 FOR K=0 TO 100:NEXTK
120 PRINT G$;SPC(9);"<SHIFT e M> <SHIFT e -> <SHIFT e
          N>"
125 PRINT SPC(10);"<SHIFT e C>*<SHIFT e C>"
130 PRINT SPC(9);"<SHIFT e N> <SHIFT e -
          > <SHIFT e M>"
135 FOR K=0 TO 100:NEXTK
140 PRINT G$;SPC(9);"  <SHIFT e ->  "
145 PRINT SPC(10);"<SHIFT e C>*<SHIFT e C>"
150 PRINT SPC(9);"  <SHIFT e ->  "
165 FOR K=0 TO 100:NEXTK
168 IF M=1 THEN 181
170 A=156:M=1:GOTO 100
181 PRINT G$;
183 PRINT SPC(6);"<CTRL e 3><COM e F><COM e F>
          <RVS ON><SHIFT e lira sterlina><COM
          e V><RVS OFF> <CTRL e 6><RVS ON><
          COM e C><COM e *><RVS OFF><COM e
          D> <COM e D>          ";
185 PRINT "<CTRL e 3><COM e D> <COM e D
          > <CTRL e 6><COM e D> <COM e D>"
190 PRINT SPC(6);" <CTRL e 3> <COM e I><COM
          e D>  <CTRL e 6><COM e F><COM e D>
          <CTRL e 7>"
195 PRINT "                    "
199 GET A$:IF A$="" THEN 199
200 GOTO 1
```

Le linee 1 e 3 preparano stringhe con caratteri di controllo.

La linea 5 ripulisce lo schermo.

Le linee da 10 a 35 preparano i disegni delle automobiline.

Le linee da 40 a 80 disegnano e fanno muovere le 2 automobiline, facendole partire, una dall'angolo in alto a sinistra e l'altra dall'angolo in basso a destra. Le automobiline muovendosi convergono verso il centro dello

schermo.

Le linee da 90 a 170 disegnano l'urto e lo scoppio delle due automobiline.

Le linee da 181 a 195 disegnano quello che resta delle automobiline dopo lo scontro.

La linea 199 crea un attesa: se si preme un qualunque tasto il programma ricomincia.

PROGRAMMA DEGLI OMINI CONTENTI

Per la preparazione delle stringhe con i caratteri di controllo si usa la stessa tecnica del precedente programma.

Questa volta non vi spieghiamo cosa fa il programma. Provatelo e studiatelo!

```
  1 CD$="<CRSR verticale>"
  5 DIM A$(2),B$(2),C$(2)
 10 PRINT"<SHIFT e CLR HOME>";:GOSUB 800
 11 PRINT"<CLR HOME>";
 15 S$="    ":W$="   "
 18 G$=CD$+CD$+CD$+CD$+CD$+CD$+CD$+CD$+CD$
 20 A1$="<SHIFT e M><SHIFT e Q><SHIFT e N>"
 21 B1$=" <SHIFT e Q> "
 30 A2$=" <COM e +> "
 31 B2$="<SHIFT e N><COM e +><SHIFT e M>"
 40 A3$="<SHIFT e N> <SHIFT e M>"
 41 B3$=" <COM e G><COM e G>"
 50 A$(1)=W$+A1$+S$+B1$+S$+A1$
 60 B$(1)=W$+A2$+S$+B2$+S$+A2$
 70 C$(1)=W$+A3$+S$+B3$+S$+A3$
 80 A$(2)=W$+B1$+S$+A1$+S$+B1$
 90 B$(2)=W$+B2$+S$+A2$+S$+BE$
 95 C$(2)=W$+B3$+S$+A3$+S$+B3$
 99 PRINT"<CTRL e 3>";
100 FOR K=1 TO 2
110 PRINT"<CLR HOME>";CD$;A$(K)
120 PRINT B$(K)
130 PRINT C$(K)
132 PRINT G$;SPC(10);MID$(A$(K),4,3)
134 PRINT SPC(10);MID$(B$(K),4,3)
136 PRINT SPC(10);MID$(C$(K),4,3)
140 FOR T=1 TO 150:NEXTT
150 NEXTK
160 GOTO 100
800 PRINT"<CTRL e 6>";CD$;CD$;CD$;CD$;CD$;
```

```
 810 PRINT"  <RVS ON><COM e K><RVS OFF><COM
           e U> <COM e J> <COM e J><COM
           e L> <COM e L> <COM e K><COM
           e L> <COM e L> <RVS ON><COM
           e B><RVS OFF><COM e F>"
 820 PRINT"  <RVS ON><COM e K><RVS OFF><COM
           e U> <COM e L><COM e L>  <COM
           e J><COM e J> <COM e K> <COM
           e J><COM e J><COM e D><RVS
           ON><COM e I><RVS OFF><COM e B>"
 830 PRINT"  <COM e C><RVS ON><COM e I><RVS
            OFF>  <COM e V>  <COM e C>  <COM
           e V> <COM e C> <COM e C> <COM e C>"
 999 PRINT
1000 PRINT"<CTRL e 7>";SPC(8);"<COM e I><RVS ON><
           COM e U>   "
1010 PRINT SPC(6);" <RVS ON><COM e V>       "
1020 PRINT SPC(6);"<RVS ON><COM e V>        "
1030 PRINT SPC(5);"<RVS ON><COM e K>   <RVS OFF><
           SHIFT e lira sterlina>    <RVS ON>   <RVS OF
           F><SHIFT e lira sterlina>"
1040 PRINT SPC(5);"<RVS ON><COM e H>  <COM e
           N><RVS OFF>      <RVS ON>  <RVS OFF><SHIFT e l
           ira sterlina>"
1050 PRINT SPC(5);"<RVS ON>   <RVS OFF><COM e K>"
1060 PRINT SPC(5);"<RVS ON><COM e H>  <COM
           e L><RVS OFF>     <RVS ON>  <COM e *>"
1070 PRINT SPC(5);"<RVS ON><COM e K>   <COM
           e *><RVS OFF>    <RVS ON>   <COM e *>"
1080 PRINT SPC(6);"<RVS ON><COM e F>       "
1090 PRINT SPC(7);"<COM e F>    "
1100 PRINT SPC(8);"<RVS ON><COM e I><COM e P>    "
1200 PRINT CD$;CD$;"  <RVS ON>VIC<RVS OFF>  <RVS ON>VIC
           <RVS OFF>  <RVS ON>VIC<RVS OFF>  <RVS ON>VIC
           <RVS OFF> <CLR HOME>"
1210 RETURN
```

PROGRAMMA PER GIOCARE CON LE NOTE E I COLORI

Segue la lista del programma:

```
1 CLR
2 CD$="<CRSR verticale>"
3 CU$="<SHIFT e CRSR verticale>"
4 CL$="<SHIFT e CRSR orizzontale>"
5 S4$=" 4 spazi"
```

```
  6 GOSUB 2000
  7 GET T$:IF T$="" THEN 7
  8 POKE 36879,8
 10 DIM A(30)
 20 FOR T=30 TO 1 STEP -1
 30 A(T)=INT(RND(0)*4+1)
 40 NEXT T
 50 PRINT "<SHIFT e CLR HOME>";
 60 FOR T=1 TO 18:PRINT CD$;:NEXTT
 65 PRINT"<CTRL E 3>";
 70 PRINT"  MAX.PUN.    ULT.PUN."
 80 PRINT SPC(4);MP%,SPC(3);UP%
 90 PRINT"<CLR HOME><RVS ON>PREMI UN TASTO":GET A$
 92 IF A$="" THEN 90
 97 PRINT"<CLR HOME>";S4$+S4$+S4$+S4$
 99 N=1
100 FOR B=1 TO N
110 K=A(B)
120 GOSUB 500
130 NEXT B
198 PRINT"<CLR HOME><CTRL e 4>TOCCA A TE"
200 FOR B=1 TO N
210 GET A$:IF A$="" THEN 210
211 K=0
213 IF A$="Y" THEN K=1
214 IF A$="F" THEN K=2
215 IF A$="J" THEN K=3
216 IF A$="B" THEN K=4
220 IF K=0 THEN 210
240 PRINT"<CLR HOME><CTRL e 1>";S4$;S4$;S4$
250 GOSUB 500
260 IF K<>A(B) THEN 400
270 NEXTB
280 FOR T=0 TO 400:NEXTT
300 N=N+1:GOTO 100
400 FOR T=1 TO 4
410 POKE 36879,24
420 POKE 36874,150
430 FOR GR=0 TO 300:NEXTGR
435 POKE 36874,0
440 POKE 36879,8
445 FOR GR=0 TO 100:NEXTGR
450 NEXTT
460 UP%=N-1
470 IF MP%<UP% THEN MP%=UP%
480 GOTO 20
500 ON K GOTO 510,530,550,570
510 PRINT"<CLR HOME>";CD$;CD$;CD$;"<CTRL e 2>";
         SPC(9);
520 GOSUB 1000:RETURN
530 PRINT"<CLR HOME>";CD$;CD$;CD$;CD$;CD$;CD$;CD$;
         CD$;"<CTRL e 8>";SPC(4);
```

```
 540 GOSUB 1000: RETURN
 550 PRINT"<CLR HOME>";CD$;CD$;CD$;CD$;CD$;CD$;CD$;
          CD$;"<CTRL e 6>";SPC(14);
 560 GOSUB 1000: RETURN
 570 PRINT"<CLR HOME>";:FOR T=1 TO 13:PRINTCD$;:NEXTT
 575 PRINT"<CTRL e 3>";SPC(9);
 580 GOSUB 1000: RETURN
1000 PRINT"<RVS ON>";S4$;SPC(18);
1100 PRINT S4$;SPC(18);S4$;SPC(18);S4$;
1350 POKE 36878,15
1400 POKE 36875,230+K
1410 FOR T=0 TO 200-5*N:NEXTT
1420 PRINT CU$;CU$;CU$;CL$;CL$;CL$;CL$;
1430 PRINT"<RVS OFF>";S4$;SPC(18);S4$;SPC(18);
          S4$;SPC(18);S4$
1470 POKE 36875,0
1480 POKE 36878,0
1490 RETURN
2000 PRINT"<SHIFT e CLR HOME>";CD$;CD$;CD$;CD$;
2010 PRINT"<RVS ON><CTRL e 4>  GIOCO DELLE NOTE   ";
2020 PRINT S4$;" E DEI COLORI ";S4$
2030 PRINT CD$;CD$;S4$;"   <RVS ON><CTRL e 1> Y=ALTO "
2040 PRINT CD$;CD$;" <RVS ON><CTRL e 8> F=SINS.";
           "<RVS OFF> <RVS ON><CTRL e 6> J=DESTRA "
2050 PRINT CD$;CD$;S4$;"  <RVS ON><CTRL e 3> ";
           "B=BASSO <RVS OFF><CTRL e 7>"
2060 RETURN
```

Il gioco consiste nel riprodurre una sequenza di suoni e colori proposta dal calcolatore. Allo scopo vengono utilizzati 4 tasti situati al centro della tastiera in croce; essi sono: <Y>, <F>, <J>, <B>.

Le regole del gioco sono le seguenti:

. Il calcolatore propone un suono (che inizialmente e' formato da una sola nota) facendo apparire contemporaneamente un quadrato colorato in una delle 4 posizioni possibili sullo schermo: alto, basso, destra, sinistra.

. Esso resta in attesa di una risposta da parte del giocatore il quale la puo' dare usando uno dei 4 tasti sopra indicati. I tasti hanno questo significato:

Y significa alto;
B   "   basso;
F   "   sinistra;
J   "   destra.

. Se la risposta del giocatore e' giusta il calcolatore aggiunge alla sequenza di note in essere un'altra nota, ed

aumenta la velocita' con la quale fa apparire i quadrati colorati.

. Se la risposta e' sbagliata il giocatore viene avvisato dal lampeggiamento dello schermo; contemporaneamente si sente un suono continuo. Nella parte bassa dello schermo vengono visualizzati: la lunghezza in numero di note dell'ultima sequenza musicale prodotta e la lunghezza della sequenza piu' lunga ottenuta durante tutto il gioco.

. Se si preme un qualunque tasto il gioco prosegue con una nuova sequenza.

Se si vuole rendere piu' difficile il gioco, si possono introdurre le linee di programma sotto riportate:

```
   8 GOSUB 3000
  10 GET T$:IF T$="" THEN 10
  12 POKE 36879,8
  15 DIM A(30)
1410 FOR T=0 TO 200-(D+1)*N:NEXTT
3000 PRINT "<SHIFT e CLR HOME> GRADO DI DIFFICOLTA' "
3010 PRINT " DA 1 A 5"
3020 INPUT D
3030 IF D<0 OR D>5 THEN 3000
3040 RETURN
```

quelle aventi lo stesso numero di quelle gia' esistenti vanno a sostituirsi alle precedenti. Inoltre si deve cancellare la linea 7 scrivendo 7 e RETURN.

EFFETTI SONORI

Si riportano alcuni pezzi di programmi per ottenere effetti sonori. Essi possono essere introdotti nei vostri programmi. Non sono stati scritti i numeri di linea per evitare confusione; potrete attribuire voi ad ogni linea il numero che concorda con il vostro programma. Tenete presente che quando vengono eseguiti questi effetti sonori il programma non fa altro e quindi fate attenzione se volete introdurli in programmi di animazione.

1) SCALE MUSICALI

```
POKE 36878,15
FOR L=250 TO 200 STEP-2
POKE 36876,L
FOR M=1 TO 100
NEXTM
NEXTL
FOR L=205 TO 250 STEP2
POKE 36876,L
FOR M-1 TO 100
NEXTM
NEXTL
POKE 36876,0
POKE 36878,0
```

2) ALLARME ROSSO

```
POKE 36878,15
FOR L=1 TO 10
FOR M=180 TO 235 STEP2
POKE 36876, M
FOR N=1 TO 10
NEXTN
NEXTM
POKE 36876,0
FOR M=1 TO 100
NEXTM
NEXTL
POKE36878,0
```

3) SUONI DA CALCOLATORE

```
POKE 36878,15
FOR L=1 TO 100
POKE 36876,INT(RND(1)*128)+
     128
FOR M=1 TO 10
NEXTM
NEXTL
POKE 36876,0
POKE 36878,0
```

4) RAGGIO LASER

```
POKE 36878,15
FOR L=1 TO 30
FOR M=250 TO 240 STEP-1
POKE 36876,M
NEXTM
FOR M=240 TO 250
POKE 36876,M
NEXTM
POKE 36876,0
NEXTL
POKE 36878,0
```

5) ESPLOSIONE

```
POKE 36877,220
FOR L=15 TO 0 STEP-1
POKE 36878,L
FOR M=1 TO 300
NEXTM
NEXTL
POKE 36877,0
POKE 36878,0
```

6) SIRENA

```
POKE 36878,15
FOR L=1 TO 10
POKE 36875,200
FOR M=1 TO 500
NEXTM
POKE 36875,0
POKE 36876,200
FOR M=1 TO 500
NEXTM
POKE 36876,0
NEXTL
POKE 36878,0
```

7) BOMBE

```
POKE 36878,10
FOR L=230 TO 128 STEP-1
POKE 36876,L
FOR M=1 TO 20
NEXTM
NEXTL
POKE 36876,0
POKE 36877,200
FOR L=15 TO 0 STEP-0.05
POKE 36878,L
NEXTL
POKE 36877,0
```

8) SEGNALE DI OCCUPATO

```
POKE 36878,15
FOR L=1 TO 15
POKE 36876,160
FOR M=1 TO 400
NEXTM
POKE 36876,0
FOR M=1 TO 400
NEXTM
NEXTL
POKE 36878,0
```

9) SQUILLO DEL TELEFONO

```
POKE 36878,15
FOR L=1 TO 5
FOR M=1 TO 50
POKE 36876,200
FOR N=1 TO 5
NEXTN
POKE 36876,0
NEXTM
FOR M=1 TO 3000
NEXTM
NEXTL
POKE 36878,0
```

10) ONDE DELL'OCEANO

```
POKE 36877,180
FOR L=1 TO 10
D=INT(RND(1)*5)*50+50
FOR M=3 TO 15
POKE 36878,M
FOR N=1 TO D
NEXTN
NEXTM
FOR M=15 TO 3 STEP-1
POKE 36878,M
FOR N=1 TO D
NEXTN
NEXTM
NEXTL
POKE 36878,0
POKE 36877,0
```

11) CINGUETTIO

```
POKE 36878,15
FOR L=1 TO 20
FOR M=254 TO 240 +
    INT(RND(1)*10) STEP-1
POKE 36876,M
NEXTM
POKE 36876,0
FOR M=0 TO INT(RND(1)*100)+
    120
NEXTM
NEXTL
```

12) UFO CHE SPARISCE

```
POKE 36878,15
FOR L=130 TO 254
POKE 36876,L
FOR M=1 TO 40
NEXTM
NEXTL
POKE 36878,0
POKE 36876,0
```

13) VENTO

```
POKE 36878,15
POKE 36874,170
POKE 36877,240
FOR L=1 TO 2000
NEXTL
POKE 36874,0
POKE 36877,0
POKE 36878,0
```

14) UFO CHE ATTERRA

```
POKE 36878,15
FOR L=1 TO 20
FOR M=220-L TO 160-L STEP-4
POKE 36876,M
NEXTM
FOR M=160-L TO 220-L STEP4
POKE 36876,M
NEXTM
NEXTL
POKE 36878,0
POKE 36876,0
```

15) UFO CHE SPARA

```
POKE 36878,15
FOR L=1 TO 15
FOR M=200 TO 220+L*2
POKE 36876,M
NEXTM
NEXTL
POKE 36878,0
POKE 36876,0
```

16) ULULATO

```
POKE 36878,15
FOR L=148 TO 200 STEP7
POKE 36876,L
NEXTL
FOR L=128 TO 200
POKE 36876,L
NEXTL
FOR L=200 TO 128 STEP-1
POKE 36876,L
NEXTL
POKE 36878,0
POKE 36876,0
```

17) SCALPITIO

```
POKE 36878,15
FOR L=1 TO 10
POKE 36874,200
FOR M=1 TO 10
NEXTM
POKE 36874,0
FOR M=1 TO 100
NEXTM
NEXTL
POKE 36878,0
```

18) CIGOLIO DI PORTA

```
POKE 36878,15
B=0
FOR L=128 TO 255 STEP11
POKE 36874,L
FOR M=1 TO 10
NEXTM
B=B+1
IF B=3 THEN B=0:POKE 36874,0
NEXTL
POKE 36874,0
POKE 36878,0
```

19) TICK TOCK

```
POKE 36878,15
FOR L=1 TO 10
POKE 36875,200
FOR M=1 TO 10
NEXTM
POKE 36875,0
FOR M=1 TO 300
NEXTM
POKE 36874,200
FOR M=1 TO 10
NEXTM
POKE 36874,0
FOR M=1 TO 300
NEXTM
NEXTL
POKE 36878,0
```

20) BIP BIP

```
POKE 36878,15
POKE 36876,220
FOR L=1 TO 5
NEXTL
POKE 36876,0
FOR L=1 TO 500
NEXTL
POKE 36876,200
FOR L=1 TO 5
NEXTL
POKE 36876,0
FOR L=1 TO 500
NEXTL
POKE 36878,0
```

Altri libri editi dal Gruppo Editoriale Jackson

Esperimenti su circuiti logici e di memoria TTL
Codice 001A — L. 22.000

Esperimenti su circuiti logici e di memoria TTL
Codice 002A — L. 22.000

IL 3UGBOOK II - Esperimenti di interfacciamento e trasmissione dati
Codice 021A — L. 4.500

IL BUGBOOK III - Interfacciamento e programmazione del microcomputer 8080
Codice 003A — L. 19.000

Interfacciamento di microcomputer Esperimenti utilizzanti Chip 8255 PP I
Codice 004A — L. 10.500

Esperimenti con TTL e 8080A, Vol. 1
Codice 005A — L. 19.000

Esperimenti con TTL e 8080A, Vol. 2
Codice 006A — L. 19.000

IL BUGBOOK VII - L'interfacciamento tra microcomputer e convertitori analogici
Codice 007A — L. 15.000

Corso di elettronica fondamentale con esperimenti
Codice 201A — L. 15.000

Comprendere l'elettronica a stato solido
Codice 202A — L. 14.000

Introduzione pratica all'impiego dei circuiti integrati digitali
Codice 203A — L. 7.000

Elettronica integrata diqitale
Codice 204A — L. 34.500

SC/MP
Codice 301D — L. 9.500

Lessico dei microprocessori
Codice 302P — L. 3.500

Introduzione al personal e business computing
Codice 303D — L. 14.000

Introduzione ai microcomputer - Il libro dei principianti, Vol. 0
Codice 304A — L. 14.000

Introduzione ai microcomputer - Il libro dei concetti fondamentali, Vol. 1
Codice 305A — L. 16.000

Practical microprocessors
Codice 308B — L. 35.000

Principi e tecniche di elaborazione dati
Codice 309A — L. 15.000

NANOBOOK Z80, Vol. 1 - Tecniche di programmazione
Codice 310P — L. 15.000

NANOBOOK Z80, Vol. 3 - Tecniche di interfacciamento
Codice 312P — L. 18.000

DBUG
Codice 313P — L. 6.000

Tecniche di interfacciamento dei microprocessori
Codice 314P — L. 22.000

Microelettronica: la nuova rivoluzione industriale
Codice 315P — L. 9.000

Elementi di trasmissione dati
Codice 316D — L. 9.000

Impariamo a programmare in Basic con lo ZX80
Codice 317B — L. 4.500

I microprocessori
Codice 320P — L. 22.000

La programmazione dello Z8000
Codice 321D — L. 22.000

TEA, un editor assembler residente per l'8080/8085
Codice 322P — L. 12.000

8080A/8085 - Programmazione in linguaggio assembly
Codice 323P — L. 24.000

Programmazione dello Z80 e progettazione logica
Codice 324P — L. 19.000

Programmazione dell'8080 e progettazione logica
Codice 325P — L. 16.500

Z80 programmazione in linguaggio assembly
Codice 326P — L. 29.500

Usare il microprocessore
Codice 327A — L. 15.000

Pascal - Manuale e standard del linguaggio
Codice 500P — L. 10.000

Impariamo il Pascal
Codice 501A — L. 10.000

Introduzione al Basic
Codice 502A — L. 18.500

Applicazioni del 6502
Codice 504B — L. 13.500

Impariamo a programmare in Basic con il PET/CBM
Codice 506A — L. 10.000

Impariamo a programmare in Basic con il VIC/CBM
Codice 507A — L. 11.000

Il Timer 555
Codice 601B — L. 8.600

La progettazione dei circuiti amplificatori operazionali con esperimenti
Codice 602B — L. 15.000

La progettazione dei filtri attivi con esperimenti
Codice 603B — L. 15.000

La progettazione dei circuiti PLL con esperimenti
Codice 604H — L. 14.000

Guida ai CMOS con esperimenti
Codice 605B — L. 15.000

I tiristori - 110 progetti pratici
Codice 606D — L. 8.000

Guida mondiale dei transistors
Codice 607H — L. 20.000

Guida mondiale degli amplificatori operazionali
Codice 608H — L. 15.000

Guida mondiale dei transistori ad effetto di campo JFET e MOS
Codice 609H — L. 10.000

Amplificatori di Norton
Codice 610B — L. 22.000

Manuale pratico del riparatore radio-TV
Codice 701P — L. 18.500

Audio Handbook
Codice 702H — L. 9.500

Audio e Hi-Fi
Codice 703D — L. 6.000

Potete acquistare i suddetti libri nelle migliori librerie oppure utilizzando il tagliando d'ordine sul retro.

Tagliando ordine **libri Jackson** da inviare a:
Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. **Città**

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

☐ Pagherò al postino l'importo di L. .... + L. 1.500 per contributo fisso spese di spedizione

☐ Allego assegno n° .... di L. ....
(in questo caso la spedizione è gratuita)

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

Tagliando ordine **libri Jackson** da inviare a:
Gruppo Editoriale Jackson - Via Rosellini, 12 - 20124 Milano

Nome Cognome

Indirizzo

Cap. **Città**

Codice Fiscale (indispensabile per le aziende)

Inviatemi i seguenti libri:

☐ Pagherò al postino l'importo di L. .... + L. 1.500 per contributo fisso spese di spedizione

☐ Allego assegno n° .... di L. ....
(in questo caso la spedizione è gratuita)

Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità	Codice Libro	Quantità

L. 12.500

Cod. 507 A

La dr. Rita Bonelli, laureata in Matematica e Fisica presso l'Università di Milano, può vantare un'esperienza di circa 25 anni nell'analisi dei sistemi organizzativi e nella programmazione dei calcolatori elettronici.
Iniziata la sua attività come venditore, ha poi lavorato sino al 1960, come analista-programmatore, sul primo calcolatore elettronico IBM installato in Italia presso un'industria.
Nei successivi 20 anni ha lavorato su medi e grandi sistemi elettronici della seconda e della terza generazione, svolgendo una intensa attività di consulenza professionale per fabbricanti di calcolatori, grandi aziende industriali e case di software.
Da più di dieci anni affianca alle attività professionali le attività didattiche, come titolare di una cattedra di informatica presso l'Istituto Tecnico Industriale Feltrinelli di Milano.
Attualmente si interessa anche di mini e personal computers, studiando il software applicativo per particolari categorie di utenti, e tenendo corsi di programmazione a vari livelli.

49
Vogliamo
Incominciare
Cosí ?
IMPARIAMO A PROGRAMMARE
IN BASIC CON IL VIC™/CBM™
R.Bonelli
GRUPPO
EDITORIALE
JACKSON